NEWSLETTER

# EUUG

European UNIX® systems User Group

Volume 9, No. 2
Summer 1989

## CONTENTS

# EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

EUUG

*Volume 9, Number 2*
*Summer*

**Editorial Team:**

| | |
|---|---|
| Philip Peake<br>Publications Executive | Axis International<br>135 Rue d'Aguesseau<br>92100 Bonlogne<br>France<br>philip@axis.fr |
| Alain Williams<br>Editor | Parliament Hill Computers Ltd.<br>7 Prospect Street<br>Caversham<br>Berkshire RG4 8JB<br>U.K.<br>addw@phcomp.co.uk |

**Typesetting:**

| | |
|---|---|
| Laura Dekker | The Instruction Set Ltd.<br>City House<br>190 City Road<br>London EC1V 2QH<br>U.K.<br>laura@inset.co.uk |
| **Printed by:** | Rank Xerox (Copy Bureaux) Ltd.<br>68-74 Rochester Place<br>London NW1 9JX<br>U.K. |
| **Published by:** | The EUUG<br>Owles Hall<br>Buntingford<br>Hertfordshire SG9 9PL<br>U.K.<br>euug@inset.co.uk (. . . !mcvax!ukc!inset!euug) |

ISSN 1011-4211

# Editorial

*Alain D. D. Williams*
*addw@phcomp.co.uk*

*Parliament Hill Computers Ltd*
*Berkshire RG4 8JB*
*UK*
*+44 734 461232*

## Brussels

The 400 of you that turned up at the recent conference in Brussels helped contribute to another successful EUUG conference. Read what Marc Gumbold has to say about it.

You will find Tom Killian's interesting paper on Mk in this newsletter, as it was not printed in the conference proceedings.

You will find announcements for the conference in Vienna this autumn. Time to start planning for what promises to be a great event.

## European Standards

This issue contains two papers that take existing products or standards and take them further by putting forward a European viewpoint and proposing a solution to our own particular problems.

This is an issue which concerns us greatly, although the USA may seem to dominate European issues can no longer be ignored. I have invited Keld and Bjarne to share their thoughts on the Europeanisation of C with you. Keld has also contributed to a paper on European Troff.

Advances on both these fronts will be reported in future issues of this newsletter.

## Regular Columns

By all accounts Donal Daly has generated a lot of interest with his review of public domain software. Many of you have asked him how to get hold of it, this he starts to explain.

Other columns keep you abreast with network developments, Unix User group events (both in Europe and the USA), standards work, or help you better use the Unix system: Colston Sanger, through Sam Nelson, has opened my eyes on some aspects of a subject that I thought I knew well.

## EUUG Publications

You will find details of various publications which have been produced by the EUUG or sister organisations.

There is more than just past copies of proceedings, but really useful references like the e-mail directory and the 4.3 manuals.

## Future Copy Dates

Readers are encouraged to submit papers for publication in the newsletter. Please contact me to talk about any ideas that you may have.

The next copy and publication dates are:

| | | |
|---|---|---|
| 24 July | for | 1 September |
| 23 October | for | 1 December |

## Advertising

Why not advertise your products and services in the EUUG Newsletter. By so doing your company will be brought to the attention of UNIX professionals throughout Europe.

You may either give us copy to be printed as part of the newsletter, you will need to supply us with electronic or camera ready copy. You may print it yourself and have it put into the envelope with the newsletter.

To have a page printed will cost you £300 – extra colours £100 each. An envelope inserts cost about the same, but may depend on the weight of your material. Non members of the EUUG pay an extra 50%.

Contact either myself or Owles Hall for more details. The copy date is the same as for papers.

# An extension to the troff character set for Europe

*E.G. Keizer*
*K.J. Simonsen*
*J. Akkerhuis*

*Vrije Universiteit, Amsterdam, The Netherlands*
*University of Copenhagen, Copenhagen, Denmark*
*C.M.U., USA*

## ABSTRACT

The typesettting program *troff* was originally written for formatting English text for the CAT 48 Typesetter. Its offspring is used for formatting a variety of languages with a large diversity of output devices. The authors agreed on an addition to the troff character set covering old and new national and international latin-based character sets.

## The problems

When adapting the UNIX† typesetting program *troff*[1] to a new output device, one wants to have access to the extra characters offered by the device, without sacrificing any characters already in use. Device-independent *troff*, also called *titroff* or *ditroff*[2] uses a flexible font definition mechanism that allows addition and deletion of characters.

Many people, including the authors, have used this mechanism[3] to add characters. This has led to a diversity of names, with the expected conflicts of using the same name for different characters and different names for the same character in different implementations. Thus *troff* input files are becoming less and less portable, even for the same output device on different installations. We regret this development and, during a conference in Copenhagen, we decided to make an attempt at some standardisation. B.W. Kernighan, author of ditroff, agreed to our proposal of acting as a clearing house for our new names and he still has to give his blessing to this article.

We realise that it is impossible to name every printable character in the world. The total amount of different characters is simply too huge. Naming all the hundred different turtles in a turtle font is both frustrating and futile. We restricted ourselves to the following categories:

- characters belonging to the printed language of several Western-European countries: Æ ß
- variations of letters, especially with accents: ï û ö
- often used mathematical symbols: $\wedge$ $\vee$ $\otimes$

## Natural language support

The *troff* character set is based on the US-ASCII standard. This standard is well suited to English text, but causes problems when used in most European countries. US-ASCII is a version of the ISO 646-1983 standard. The ISO standard states the characters used in 7-bit ASCII and contains 94 printable graphic symbols. ISO-646 allows national versions for 12 of its character positions. The European Computer Manufacturers' Association ECMA is registering all different national versions of ISO 646-1983 and assigns

† UNIX is a trademark of Bell Laboratories.

a different character to each. This allows the creation of documents with multiple character sets. The assigned character serves to identify each character set in such doucuments. The table below shows several versions conforming to ISO 646.

| National ISO 646 character sets | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Country | Standard | .la parameter | | | | | | | | | | | | | |
| ISO | ISO 646 IRV | ISO | # | ¤ | @ | [ | \ | ] | ^ | ‘ | { | \| | } | ‾ |
| USA | X3.4-1968 | US | # | $ | @ | [ | \ | ] | ^ | ‘ | { | \| | } | ˜ |
| Great Britain | BS 4730 | GB | £ | $ | @ | [ | \ | ] | ^ | ‘ | { | \| | } | ‾ |
| Japan | JIS C 6229 | JP | # | $ | @ | [ | ¥ | ] | ^ | ‘ | { | \| | } | ‾ |
| China | GB 1988-80 | CN | # | ¥ | @ | [ | \ | ] | ^ | ‘ | { | \| | } | ‾ |
| Denmark | DS 2089 | DK | # | $ | @ | Æ | Ø | Å | ^ | ‘ | æ | ø | å | ˜ |
| Norway | NS 4551-1 | NO | # | $ | @ | Æ | Ø | Å | ^ | ‘ | æ | ø | å | ‾ |
| | NS 4551-2 | NO2 | § | $ | @ | Æ | Ø | Å | ^ | ‘ | æ | ø | å | \| |
| Finland | | FI | # | $ | @ | Ä | Ö | Å | ^ | ‘ | ä | ö | å | ˜ |
| Sweden | SEN 850200 B | SE | # | ¤ | @ | Ä | Ö | Å | Ü | ‘ | ä | ö | å | ‾ |
| | SEN 850200 C | SE2 | # | ¤ | É | Ä | Ö | Å | Ü | é | ä | ö | å | ü |
| Germany | DIN 66 003 | DE | # | $ | § | Ä | Ö | Ü | ^ | ‘ | ä | ö | ü | ß |
| Hungary | MSZ 7795/3 | HU | # | ¤ | Á | É | Ö | Ü | ^ | á | é | ö | ü | ″ |
| France | NF Z 62-010 | FR | £ | $ | à | ° | ç | § | ^ | µ | é | ù | è | ¨ |
| Italy | | IT | £ | $ | § | ° | ç | é | ^ | ù | à | ò | è | ì |
| Spain | | ES | £ | $ | § | ¡ | Ñ | ¿ | ^ | ‘ | ° | ñ | ç | ˜ |
| | | ES2 | # | $ | • | ¡ | Ñ | Ç | ¿ | ‘ | ´ | ñ | ç | ¨ |
| Portugal | | PT | # | $ | § | Ã | Ç | Õ | ^ | ‘ | ã | ç | õ | ° |
| | | PT2 | # | $ | ´ | Ã | Ç | Õ | ^ | ‘ | ã | ç | õ | ˜ |

Terminals in these countries are often adapted to these national variations. Creating *troff* input for Danish texts on Danish terminals is a frustrating experience. One has to type **\(AE** for Æ in spite of the presence of a special key for Æ. You can by-pass this by using the *troff* command **.tr**, which allows the mapping of any character to any other. We have employed the **.tr** command in a macro **.la** which is designed to make it possible to shift between all the ISO 646 input character sets in the above table. The macro takes a code for the country as parameter; the first two letters being the ISO 3166 two-letter country code. The **.la** macro can be used like this:

```
.la US
First we write something in "God's own" character set.
.la DK
S} skriver vi noget s|dt p} dansk: sodavandsis.
.la DE
Und f}r Deutschen k|nnen wir auch etwas schreiben!
.la US
```

giving:

First we write something in "God's own" character set.
Så skriver vi noget sødt på dansk: sodavandsis.
Und für Deutschen können wir auch etwas schreiben!

The **.la** macro can only be used when all the characters of the character sets in use have a unique code on the printing device. Also you cannot change input character set within a diversion in *troff*, you need to use the special character names if you want to use foreign characters within a sentence. An example of having a French name in a Danish text:

```
    Jeg s} Jer\(^ome Fran\(,cois komme til K|benhavn.
```

giving:

Jeg så Jerôme François komme til København.

To be able to use all other national characters within a national character set, we decided to introduce names for all the different national characters, even for the 'default' US names.

Also we went through the new ISO standards for Latin alphabets (ISO 8859) and assured that all special characters there would have a unique name according to this proposal.

A last warning is about the *troff* escape character \ and the national characters taking its place. Here you must write the national character followed by an 'e' to get the desired result.

The **.la** macro has the following contents:

```
.de la
.\" languages - keld@dkuug.dk & storm@dkuug.dk
.\" Covers all ECMA registrered versions of ISO 646
.\" Countries according to ISO 3166
.\" Commented is registered ECMA char code and standard number
.fl
.ie \\n(.$=0 .ds )L \\*(=L
.el .ds )L \\$1
.ds =L \\*(LA
.ds LA \\*()L
.rm )L
.if "\\*(LA"DK"  .tr #\(sh$\(Do@\(at[\(AE\\\\\(/O]\(oA^\(ha\'\(ga{\(ae|\(/o}\(oa~\(ti  \"   DS 2089
.if "\\*(LA"US"  .tr #\(sh$\(Do@\(at[\(1B\\\\\(rs]\(rB^\(ha\'\(ga{\(1C|\(ba}\(rC~\(ti  \" B X3.4-1968
.if "\\*(LA"ISO" .tr #\(sh$\(Cs@\(at[\(1B\\\\\(rs]\(rB^\(ha\'\(ga{\(1C|\(ba}\(rC~\(rn  \" @ IRV
.if "\\*(LA"GB"  .tr #\(Po$\(Do@\(at[\(1B\\\\\(rs]\(rB^\(ha\'\(ga{\(1C|\(ba}\(rC~\(rn  \" A BS 4730
.if "\\*(LA"DE"  .tr #\(sh$\(Do@\(sc[\(:A\\\\\(:O]\(:U^\(ha\'\(ga{\(:a|\(:o}\(:u~\(ss  \" K DIN 66 003
.if "\\*(LA"FR"  .tr #\(Po$\(Do@\('a[\(de\\\\\(,c]\(sc^\(ha\'\(mu{\('e|\('u}\('e~\(ad  \" f NF Z 62-010 (1982)
.if "\\*(LA"CN"  .tr #\(sh$\(Ye@\(at[\(1B\\\\\(rs]\(rB^\(ha\'\(ga{\(1C|\(ba}\(rC~\(rn  \" T GB 1988-80
.if "\\*(LA"JP"  .tr #\(sh$\(Do@\(at[\(1B\\\\\(Ye]\(rB^\(ha\'\(ga{\(1C|\(ba}\(rC~\(rn  \" n JIS C 6229-1984
.if "\\*(LA"IT"  .tr #\(Po$\(Do@\(sc[\(de\\\\\(,c]\('e^\(ha\'\('u{\('a|\('o}\('e~\('i  \" Y
.if "\\*(LA"ES"  .tr #\(Po$\(Do@\(sc[\(r!\\\\\(~N]\(r?^\(ha\'\(ga{\(de|\(~n}\(,c~\(ti  \" Z
.if "\\*(LA"ES2" .tr #\(sh$\(Do@\(bu[\(r!\\\\\(~N]\(,C^\(r?\'\(ga{\(aa|\(~n}\(,c~\(ad  \" h
.if "\\*(LA"PT"  .tr #\(sh$\(Do@\(sc[\(~A\\\\\(,C]\(~O^\(ha\'\(ga{\(~a|\(,c}\(~o~\(de  \" L
.if "\\*(LA"PT2" .tr #\(sh$\(Do@\(aa[\(~A\\\\\(,C]\(~O^\(ha\'\(ga{\(~a|\(,c}\(~o~\(ti  \" g
.if "\\*(LA"HU"  .tr #\(sh$\(Cs@\('A[\('E\\\\\(:O]\(:U^\(ha\'\('a{\('e|\(:o}\(:u~\(a"  \" i MSZ 7795/3
.if "\\*(LA"NO"  .tr #\(sh$\(Do@\(at[\(AE\\\\\(/O]\(oA^\(ha\'\(ga{\(ae|\(/o}\(oa~\(rn  \" ' NS 4551 - 1
.if "\\*(LA"NO2" .tr #\(sc$\(Do@\(at[\(AE\\\\\(/O]\(oA^\(ha\'\(ga{\(ae|\(/o}\(oa~\(bv  \" a NS 4551 - 2
.if "\\*(LA"SE"  .tr #\(sh$\(Cs@\(at[\(:A\\\\\(:O]\(oA^\(ha\'\(ga{\(:a|\(:o}\(oa~\(rn  \" G SEN 850200 B
.if "\\*(LA"SE2" .tr #\(sh$\(Cs@\('E[\(:A\\\\\(:O]\(oA^\(:U\'\('e{\(:a|\(:o}\(oa~\(:u  \" H SEN 850200 C
.if "\\*(LA"FI"  .tr #\(sh$\(Do@\(at[\(:A\\\\\(:O]\(oA^\(ha\'\(ga{\(:a|\(:o}\(oa~\(ti
.if "\\*(LA"JS"  .tr #\(sh$\(Do@\(vZ[\(vS\\\\\(-D]\('C^\(vC\'\(vz{\(vs|\(-d}\(vc~\('c  \" z JUS I.B1.002
..
```

## Problems we did not pursue

Written English is based on the latin alphabet, it hardly uses accents and other variations of letters. Other languages make use of accents above (è), below (ç) and through (ø) the letters. We do not address the problems of languages with more than one accent per letter and accents connecting letters. In our opinion these problems have to be solved by separate preprocessors. This allows a much more friendly user interface. These preprocessors could also solve the problem of hyphenation and ligatures for these languages.

## The troff naming scheme

*Troff* has three ways of naming characters:

- one-character ASCII names like **A** for A, **B** for B and **@** for @.
- escaped one-character names prefixed by a \ like \– for current font minus, \e for backslash and \´ for acute accent. There are only a few of these.
- two-character names prefixed by the indicator \( like \(**sc** for §, \(***g** for γ and \(**14** for ¼.

The sets of one-character and escaped one-character names are fixed. Only the set of two-character names can be extended.

## Choosing new names

While choosing names for new characters we were very much aware of the fact that the restriction of two characters per name defies all attempts to choose a consistent and logical naming scheme. Still we used a few principles in choosing the new names. Whenever these principles conflicted we refrained from long discussions but placed more value on a quick decision.

Our principles:

- may not conflict with original troff manual[1]
- try to avoid the national characters: # $ @ [ \ ] ^ ' { | } ~
- use names associated with the graphical description of the symbol. Thus \(**oA** for Å instead of \(**AA**.
- Whenever a character is a combination of an accent and a letter use as name a character representing the accent followed by the letter. For example: \(**'e** for é. This is the way these combinations were made on old-fashioned typewriters: first hit the dead key with the accent then the key with the letter. The symbols we used for the accents (and the like) are:

| ASCII | ´ | ` | : | , | ~ | o | ^ | . | " | u | v | - | / |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Accent | ´ | ` | ¨ | ¸ | ˜ | ˚ | ˆ | ˙ | ˝ | ˘ | ˇ | - | / |
| Name | aa | ga | ad | ac | a˜ | ao | aˆ | a. | a" | ab | ah | a- | / |

- fractions are done in the natural way: \(*nm* for $^n/_m$.
- Currency signs have two letter names. The first letter is Capital, the second small.
- The names for accents start with an **a**, for example \(**ad** for ¨.

## The weird characters one has

Suppose you have a *flat*, which is unlikely on any other device, but still want to have your input sent to your friend/editor which can run it on his troff. The first thing to do is not using the character as it is implemented on your troff (f.i. \(ft), but define a string at the beginning of the text and use that. The next thing to do is to give a description how it should look like. So if we stick with our flat, you end up at the start of the your input file with:

```
.\" We use a "flat" (\(ft) a lot instead of a backslash, because it
.\" stands out nicely and, since a backslash can be interpreted in
.\" so many ways, I want to make clear that you see an escape when I
.\" talk about it....
.\" So I define the string ft
.ds ft \(ft
.\" A "flat" is what in music stands for lowering the current
.\" tone. If that not is clear, consider a | (pipe character) with
.\" a small circle attached to it on the left bottom side.
.\" A define in the style of
.\"     .ds \o'|o'
.\" will do if you don't have anything more than a lineprinter around.
.\" If you want to be really fancy, you might want to try
.\" something like:
.\" .nr x \w'o'/2u
.\" .nr y .2m
.\" .ds ft \v'-\nyu'\z|\h'\nxu'\v'\nyu'\S'-9'o\S'0'
.\" Fancy ain't it?
```

So the rest of your article will look (at the input side) as:

```
An escape (denoted by \*(ft) in troff will introduce a two
character name XX by \*(ft( (so \*(ft(XX) and a two character
named QQ string interpolation will be triggered
by \*(ft*( (\*(ft*(QQ).
```

## The future

New versions of *troff* include more forms of naming characters. Both \C*'arbitrary long name'* and \N*'absolute Number of character in current font'* are used. The first allows character names of arbitrary length. This does not solve the problems discussed in this article. Worse, it will even be harder to choose names upon which a sizeable number of people will agree.

## Bibliography

1. J. F. Ossanna, "NROFF/TROFF User's Manual," Comp. Sci. Tech. Rep. 54, Bell Laboratories, Murray Hill, NJ (October 1976).

2. B. W. Kernighan, "A typesetter-independent TROFF," Comp. Sci. Tech. Rep. 97, Bell Laboratories, Murray Hill, NJ (March 1982).

3. Mark Kahrs and Lee Moore, "Adventures with Typesetter-Independent TROFF," TR 159, University of Rochester, NY (June, 1985).

## Our character set

We divided the characters in four categories. Each character is mentioned only once. Whenever we doubted we tried to place a character in the category we thought was most suitable. Lastly we included the characters from Ossanna's troff document for reference.

*Symbols from ISO 646 standards*

| **Char** | **Name** | | **Char** | **Name** | |
|---|---|---|---|---|---|
| # | \(sh | sharp | ¥ | \(Ye | Yen |
| ¤ | \(Cs | Currency sign | $ | \(Do | Dollar |
| £ | \(Po | English pound | @ | \(at | at sign |
| [ | \(lB | left square bracket | \ | \(rs | backslash |
| ] | \(rB | right square bracket | ^ | \(ha | hat or accent circumflex |
| { | \(lC | left curly bracket | } | \(rC | right curly bracket |
| \| | \(ba | bar (possibly broken) | ˜ | \(ti | tilde |
| ˝ | \(a" | accent double quote | Æ | \(AE | AE |
| Ø | \(/O | O slash | Å | \(oA | A circle |
| æ | \(ae | ae | ø | \(/o | o slash |
| å | \(oa | a circle | Ä | \(:A | A diaeresis |
| Ö | \(:O | O diaeresis | Ü | \(:U | U diaeresis |
| é | \('E | e acute accent | ä | \(:a | a diaeresis |
| ö | \(:o | o diaeresis | ü | \(:u | u diaeresis |
| ß | \(ss | German ringel S | à | \(‘a | a grave accent |
| ç | \(,c | c cedilla | è | \(‘e | e grave accent |
| ù | \(‘u | u grave accent | ò | \(‘o | o grave accent |
| ì | \(‘i | i grave accent | ¡ | \(r! | reverse ! |
| Ñ | \(~N | N tilde | ¿ | \(r? | reverse ? |
| ñ | \(~n | n tilde | ã | \(~a | a tilde |
| â | \(^a | a hat | Ç | \(,C | C cedilla |
| ê | \(^e | e hat | î | \(^i | i hat |
| õ | \(~o | o tilde | û | \(^u | u hat |
| É | \('E | E acute accent | Á | \('A | A acute accent |
| á | \('a | a acute accent | í | \('i | i acute accent |
| c | \('c | c acute accent | C | \('C | C acute accent |
| Õ | \(~O | O tilde | Ã | \(~A | A tilde |
| ð | \(-d | d bar | Ð | \(-D | Capital Icelandic Eth (D)/D bar |

*Symbols from ISO 8859 standards*

| **Char** | **Name** | | **Char** | **Name** | |
|---|---|---|---|---|---|
| ð | \(Sd | small Icelandic eth (d) | ł | \(/l | Polish l |
| Ł | \(/L | Polish L | þ | \(Tp | Small Icelandic Thorn |
| Þ | \(TP | Capital Icelandic Thorn | ¦ | \(bb | broken bar |
| $^1$ | \(S1 | superscript 1 | $^2$ | \(S2 | superscript 2 |
| $^3$ | \(S3 | superscript 3 | ë | \(:e | e diaeresis |
| È | \(‘E | E grave accent | ó | \('o | o acute accent |
| ô | \(^o | o hat | ú | \('u | u acute accent |
| ª | \(Of | feminin ordinator indicator | º | \(Om | masculin ordinator indicator |
| « | \(Fo | double french open quote | » | \(Fc | double french close quote |
| ¸ | \(ac | accent cedilla | ¨ | \(ad | accent diaeresis |
| ¶ | \(ps | english paragraph sign | ï | \(:i | i diaeresis |
| ½ | \(12 | one half | ¼ | \(14 | one quart |
| ¾ | \(34 | three quart | · | \(md | centered dot |
| ¬ | \(no | not | | | |

## *Typographical characters*

| **Char** | **Name** | | **Char** | **Name** | |
|---|---|---|---|---|---|
| ˘ | \(ab | accent breve | ˚ | \(ao | accent corona |
| ˇ | \(ah | accent haček | ˙ | \(a. | accent dot |
| ˜ | \(a~ | accent tilde | ˆ | \(a^ | accent circumflex |
| ¯ | \(a- | accent macron | ˛ | \(ho | hook |
| ı | \(.i | dotless i | ȷ | \(.j | dotless j |
| ‹ | \(fo | french open quote | › | \(fc | french close quote |
| IJ | \(IJ | ligature IJ | ij | \(ij | ligature ij |
| ™ | \(tm | Trade Mark | ‘ | \(oq | open quote |
| œ | \(oe | French oe | Œ | \(OE | French OE |
| ✓ | \(OK | check mark | ‰ | \(%0 | per mille |

## *Mathematical characters*

| **Char** | **Name** | | **Char** | **Name** | |
|---|---|---|---|---|---|
| ħ | \(-h | h bar | ⊕ | \(c+ | circle plus |
| ⊗ | \(c* | circle times | ≅ | \(=~ | approximately equal |
| ∧ | \(OR | logical or | ∨ | \(AN | logical and |
| ∀ | \(fa | for all | ∃ | \(te | there exists |
| ∴ | \(3d | therefore | ⊥ | \(pp | perpendicular to |
| ∉ | \(nm | not a member | ⊃ | \(cn | contains |
| ⊅ | \(cn | does not contain | ≈ | \(~~ | approximately |
| ℵ | \(Ah | Aleph | ≢ | \(ne | not equivalent |
| ∓ | \(-+ | minus plus | | | |

## *Symbols from the Troff manual by Ossanna*

| **Char** | **Name** | | **Char** | **Name** | |
|---|---|---|---|---|---|
| — | \(em | 3/4 Em dash | - | \(hy | hyphen |
| • | \(bu | bullet | □ | \(sq | square |
| _ | \(ru | rule | ¼ | \(14 | 1/4 |
| ½ | \(12 | 1/2 | ¾ | \(34 | 3/4 |
| fi | \(fi | fi | fl | \(fl | fl |
| ffi | \(Fi | ffi | ff | \(ff | ff |
| ffl | \(Fl | ffl | ° | \(de | degree |
| † | \(dg | dagger | ′ | \(fm | foot mark |
| ¢ | \(ct | cent sign | ® | \(rg | registered |
| © | \(co | copyright | + | \(pl | math plus |
| − | \(mi | math minus | = | \(eq | math equals |
| ∗ | \(** | math star | § | \(sc | section |
| ´ | \(aa | acute accent | ` | \(ga | grave accent |
| _ | \(ul | underline | / | \(sl | slash (matching backslash) |
| α | \(*a | alpha | β | \(*b | beta |
| γ | \(*g | gamma | δ | \(*d | delta |
| ε | \(*e | epsilon | ζ | \(*z | zeta |
| η | \(*y | eta | θ | \(*h | theta |
| ι | \(*i | iota | κ | \(*k | kappa |
| λ | \(*l | lambda | μ | \(*m | mu |
| ν | \(*n | nu | ξ | \(*c | xi |

| Char | Name | | Char | Name | |
|---|---|---|---|---|---|
| ο | \(*o | omricron | π | \(*p | pi |
| ρ | \(*r | rho | σ | \(*s | sigma |
| ς | \(ts | terminal sigma | τ | \(*t | tau |
| υ | \(*u | upsilon | φ | \(*f | phi |
| χ | \(*x | chi | ψ | \(*q | psi |
| ω | \(*w | omega | Α | \(*A | Alpha |
| Β | \(*B | Beta | Γ | \(*G | Gamma |
| Δ | \(*D | Delta | Ε | \(*E | Epsilon |
| Ζ | \(*Z | Zeta | Η | \(*Y | Eta |
| Θ | \(*H | Theta | Ι | \(*I | Iota |
| Κ | \(*K | Kappa | Λ | \(*L | Lambda |
| Μ | \(*M | Mu | Ν | \(*N | Nu |
| Ξ | \(*C | Xi | Ο | \(*O | Omricron |
| Π | \(*P | Pi | Ρ | \(*R | Rho |
| Σ | \(*S | Sigma | Τ | \(*T | Tau |
| Υ | \(*U | Upsilon | Φ | \(*F | Phi |
| Χ | \(*X | Chi | Ψ | \(*Q | Psi |
| Ω | \(*W | Omega | √ | \(sr | square root |
| ‾ | \(rn | root en extender | ≥ | \(>= | >= |
| ≤ | \(<= | <= | ≡ | \(== | identical equal |
| ≈ | \(~= | approx = | ~ | \(ap | approximates |
| ≠ | \(!= | not equal | → | \(-> | right arrow |
| ← | \(<- | left arrow | ↑ | \(ua | up arrow |
| ↓ | \(da | down arrow | × | \(mu | multiply |
| ÷ | \(di | divide | ± | \(+- | plus-minus |
| ∪ | \(cu | cup (union) | ∩ | \(ca | cap (intersection) |
| ⊂ | \(sb | subset of | ⊃ | \(sp | superset of |
| ⊆ | \(ib | improper subset | ⊇ | \(ip | improper superset |
| ∞ | \(if | infinity | ∂ | \(pd | partial deriative |
| ∇ | \(gr | gradient | ¬ | \(no | not |
| ∫ | \(is | integral sign | ∝ | \(pt | proportional to |
| ∅ | \(es | empty set | ∈ | \(mo | member of |
| \| | \(br | box vertical rule | ‡ | \(dd | double dagger |
| ☞ | \(rh | right hand | ☜ | \(lh | left hand |
| ◎ | \(bs | Bell System logo | \| | \(or | or |
| ○ | \(ci | circle | ⎧ | \(lt | left top of big curly bracket |
| ⎩ | \(lb | left bottom | ⎫ | \(rt | right top |
| ⎭ | \(rb | right bottom | ⎨ | \(lk | left center |
| ⎬ | \(rk | right center | ⎪ | \(bv | bold verticel |
| ⌊ | \(lf | left floor | ⌋ | \(rf | right floor |
| ⌈ | \(lc | left ceiling | ⌉ | \(rc | right ceiling |

# Mk'ing Hardware: A Tutorial

*Tom Killian*
*tom@research.att.com*

*AT&T Bell Laboratories*
*Murray Hill, New Jersey 07974 U.S.A*

Tom Killian started life as a high-energy physicist. He worked at CERN, in Geneva, and at Brookhaven National Laboratory on Long Island. In both places he was unable to persuade his colleagues to stop writing in FORTRAN. He joined the Computing Science Research Center at Bell Labs in 1982, where his research interests include operating systems, real-time hardware and software, laser-printer hardware and software, computer music, and espresso machine repair.

Suppose we wish to build a piece of hardware, using wire-wrap technology. At the end of the design process, we will have a set of 'object' files, most likely a list of point-to-point wiring instructions for an automatic or semi-automatic wiring machine, and one or more files for configuring programmed array logic (PAL's). These 'object' files will be generated from a plethora of 'source' files, graphic as well as textual; the details depend on what programs are available. Mk is the natural tool to tie these programs together, and ensure that the computer always has an accurate representation of the physical state of the board as it is debugged and modifications are made. In contrast to the primordial `make`, mk allows meta-rules to be defined with regular expressions, and transitive closure is part of its semantics. Using examples from the UNIX™ Circuit Design System (UCDS), we show how these properties can be applied to construct a master mk 'library' that allows rules for individual designs to be specified compactly. Such a library also facilitates tracking of changes in the design system.

## Introduction

We begin with an overview of files and dataflow in the UCDS system.

The logical part of a design usually consists of circuit diagrams (`.j` files) and PAL equations (`.e` files). The circuit contains *chips*, each identified by *name* (which is arbitrary) and *type* (which is generic, e.g., `74LS74`). Chips have *pins*, each identified by a *pin name*; pin names are local to a chip, and are a property of its type. Pins are connected by *nets*, which have unique *net names* (assigned by the drawing package if omitted by the user). It is an error for a pin to be connected to more than one net. The semantics of a circuit diagram (its `.w` file) are derived from the `.j` file by listing, for each chip, its name, type, and net–pin connections.

The physical design consists of tables matching pin names with numbers for each chip type, and chip types with packages (`.pins` files), geometric descriptions of chip packages (`.pkg` files), a geometric description of the wire-wrap board (`.brd` file), and a list of positions of the chips to be placed on the board (`.pos` file). Most of the information in `.pins` and `.pkg` files comes from standard libraries, but the user must generally supply some of it, usually for I/O connectors. In the case of a PAL, pin information and package type are extracted from its `.e` file into a `.p` file.

The wire-wrap 'compilation' process proceeds as follows. First, a `.w` file is generated from each `.j` file, and a `.p` file is made from each `.e` file; next the `.w` files are read in conjunction with the `.p` files and any pin libraries to make a `.pins` file; now the `.w` and `.pins` files are processed to make the `.net` file. This step also performs macro expansions, but that need not concern us; the salient property of the `.net` file is that all nets are expressed in terms of pin numbers. Now the `.net` file is read in conjunction with package libraries to make a `.pkg` file, and the `.net`, `.pkg`, and `.brd` files are used to generate position information (`.pos` file). Finally, the `.net`, `.pos`, `.pkg`, and `.brd` files are used to make the point-to-point wrap list (`.wr` file), and the process is complete.

PAL configuration is a three-step process. First the logic equations (in the `.e` file) are transformed into a representation of functions as sums of minterms (`.min` file). Then the `.min` file is transformed into a fuse list appropriate to a particular device. Finally, the fuse list is downloaded to a PAL programmer that burns the fuses. The last step requires knowledge of the PAL manufacturer as well as it type, since different manufacturers often use different programming algorithms.

## The Simplest Case

Suppose we have a design specified by `design.j`, `control.e`, and `design.brd`, with libraries `/usr/ucds/lib/lib.pins`, and `/usr/ucds/lib/lib.pkg`. Then our mk file might appear as follows:

```
design.w:    design.j
    jraw -w design.j >design.w

control.p:    control.e
    lde -W control.e >control.p

design.pins:    design.w control.p /usr/ucds/lib/lib.pins
    mkpins design.w control.p /usr/ucds/lib/lib.pins >design.pins

design.net:    design.w design.pins
    cdmglob -f -v design.w design.pins >design.net

design.pkg:    design.net /usr/ucds/lib/lib.pkg
    mkpkg design.net /usr/ucds/lib/lib.pkg >design.pkg

design.pos:    design.net design.pkg design.brd
    place design.net design.pkg design.brd >design.pos

design.wr:    design.net design.pkg design.brd design.pos
    wrap -c -v design.net design.pkg design.brd design.pos >design.wr
```

Transitive closure guarantees the correct dependency semantics. The file can be made more generic by using meta-rules where possible, and parameterizing elsewhere. The recipes can be made less prolix by use of `$prereq` and `$target`:

```
NAME=design
PALPINS=control.p

HWLIB=/usr/ucds/lib
LIBPINS=$HWLIB/lib.pins
LIBPKG=$HWLIB/lib.pkg

wr:N:    $NAME.wr

%.w:    %.j
    jraw -w $stem.j >$target

%.p:    %.e
    lde -W $stem.e >$target

%.pins:    %.w $PALPINS $LIBPINS
    mkpins $prereq >$target

%.net:    %.w %.pins
    cdmglob -f -v $prereq >$target

%.pkg:    %.net $LIBPKG
    mkpkg $prereq >$target
```

```
%.pos:     %.net %.pkg %.brd
    place $prereq >$target

%.wr:     %.net %.pkg %.brd %.pos
    wrap -c -v $prereq >$target
```

With a little reorganisation, we can support several pieces of hardware that have this structure, each in a separate directory such as `hw/design`. In this directory we put a file called `config` containing the first two lines from the mk file. The remainder of the mk file goes into a public place, say, `/usr/ucds/lib/mkfile`. Finally, we put this one-liner into our search `PATH`, say, `/usr/ucds/bin/Mk`:

```
exec mk -f config -f /usr/ucds/lib/mkfile "$@"
```

Now the command '`Mk wr`', executed from `hw/design`, will have the effect of building `design.wr`.

## Realistic Complications

The first problem with this scheme is that most designs have several `.j` and `.e` files, so these must be specified in `config` in addition to the `NAME`:

```
NAME=design
JFILES=cpu.j io.j
EFILES=control.e decode.e
```

Now we can use backquote substitution to get the lists of `.w` and `.p` files that we need. At the beginning of our mk file, we insert:

```
ENVIRON="resuffix(){
    echo \$1 | sed 's/\.[^ \t]*/.'\$2'/g'
}"
WFILES=`resuffix "$JFILES" w`
PALPINS=`resuffix "$EFILES" p`
```

Note the use of `\$` to prevent mk from attempting parameter substitution when `ENVIRON` is assigned. The rules using `%.w` as a prerequisite must be modified to use `$WFILES`:

```
$NAME.pins:     $WFILES $PALPINS $LIBPINS
    mkpins $prereq >$target

$NAME.net:     $WFILES $NAME.pins
    cdmglob -f -v $prereq >$target
```

Another problem is with the dependency of the `.pos` file. The mk file says that positions must be recomputed whenever the `.net` file changes, but only rarely will this be the case. Since positions are most likely determined through a complex interactive process, we don't want their calculation triggered gratuitously. So, we replace the rule for `%.pos` with this one:

```
%.place:     %.net %.pkg %.brd
    place $prereq >%.pos
```

This allows the user to work on the positioning at will, since `%.place` will never be 'up-to-date'. Conversely, since `%.place` does not appear in the prerequisites for any rule, mk will never force the user to do placement unnecessarily.

## Updating the Design

The most serious problem to be addressed is that of reworking the design. Invariably changes must be made after the board has been built, and of course we don't want to rewire from scratch. The UDCS program `rework` bears the brunt of this. Given old and new `.wr` files (representing the present and desired states of the board), it produces three files as output: a list of old wires to unwrap (`UN.wr`), a list of new wires to wrap (`RE.wr`), and a file representing the new board state (`NEW.wr`). (`Rework` may alter wrap levels in order to minimise the sizes of `UN.wr` and `RE.wr`, so `NEW.wr` may not be identical to the new input file.)

It is vital that mk not overwrite the old `.wr` file until we are sure that we no longer need it, and it would be nice if it invoked `rework` when appropriate. So we change the recipe for `%.wr` as follows:

```
%.wr: %.brd %.pkg %.net %.pos
    wrap -c -v $prereq >N$target
    if [ -f $target ]; then
        rework $target N$target && mv NEW.wr N$target
    else
        mv N$target $target
    fi
```

Now we make, e.g., `Ndesign.wr`. If `design.wr` already exists, we run `rework` and install its output as the new wrap file (since we assume one design per directory, we don't bother renaming `UN.wr` and `RE.wr`); otherwise we simply rename the output of `wrap`. It is important that the old wrap file is never touched or renamed, so we can change our minds and '`Mk wr`' as often as we like before physically changing the board.

When we actually do change the board, the presence of multiple wrap files becomes a liability. So we add a mk rule to invoke when we have finished rewrapping:

```
%.commit:
    mv N$stem.wr $stem.wr && rm UN.wr RE.wr
```

## Some Extra Touches

We would like to be able simply to type '`Mk place`', '`Mk wr`', etc., instead of having always to supply the stem. Instead of adding a rule for each such abbreviation, we can use a single regular-expression meta-rule that uses back-referencing to handle them all:

```
^(place|wr|commit)\$:RN: $NAME.\\1
```

We generally need design-specific `.pins` and `.pkg` libraries, so we make the amendments:

```
LIBPINS=$HWLIB/lib.pins io.pins my.pins
LIBPKG=$HWLIB/lib.pkg my.pkg
```

In order to make these files optional, we add the rule:

```
my.pkg my.pins io.pins:
    touch $target
```

Thus empty files will be made if required, but since no dependency is specified existing files will not be touched.

Finally, we add a rule for running the wiring machine:

```
%.wrap:
    if [ -f N$stem.wr ]; then
        wrap -s -r$ORIENT UN.wr
        wrap -s -r$ORIENT RE.wr
    else
        wrap -s -r$ORIENT $stem.wr
    fi
```

This rule chooses the appropriate `.wr` file or files. `ORIENT` is a new parameter to be set in the `config` file to specify the orientation of the board in the machine, and is the principle reason for bothering with the rule.

## PAL Programming

PAL programming in UCDS suffers from some interesting quirks. Lde, the program that generates minterms, quite properly doesn't care about the properties of real devices; such knowledge is left to pal, which generates the fuse list. Thus the .min file must be fed to pal to know whether or not the equations are going to fit, but the type of PAL that we are using (e.g., 16L8) is not part of the .min file! It happens that the PAL type is easily extracted from the .p file, so given a program (paltype) to do this, we can generate and test our .min files like this:

```
%.min:      %.e %.p
        lde -a $stem.e | quine | cover | hazard >$stem.min.ng
        pal -d `paltype $stem.p` $stem.min.ng >/dev/null
        mv $stem.min.ng $target
```

First we make a .ng (no good) file. If an output has too many terms, pal will exit with non-zero status, the recipe will be aborted, and the .ng file may be studied to figure out a way around the problem. Otherwise the .ng file becomes the target. (It is not useful to retain the output of pal, since it costs little to compute and is incomprehensible to humans.)

When we actually burn a PAL, we have to specify the manufacturer. It makes little sense to save this in a file, since we don't know whose devices the stockroom will have on any given day. So we make the manufacturer name part of the target string in a regular-expression meta-rule:

```
'^([^.]+)\.drom\.([^.]+)$':R:     \\1.min \\1.p
        man=$stem2
        type=`paltype $stem1.p`
        pal -d $type $stem1.min | tee to | drom -w -t $man$type
        drom -t $man$type >from
        cmp to from; rm to from
```

Now if we type, say, Mk control.drom.TI (and control.e calls for a 16L8), the first command executed will be:

```
    pal -d 16L8 control.min | tee to | drom -w -t TI16L8
```

This writes the PAL (drom -w), and saves a copy of the fuse list. The remainder of the recipe reads back the PAL and checks that it was programmed correctly.

## Conclusions

We have used this single mkfile with multiple config files to manage about twenty different pieces of hardware. It has proved especially convenient on small designs. Also appreciated is the ability to add targets (e.g., check to perform diagnostics, draw to make plots) to the central mkfile, and have them globally available.

## References

A. G. Fraser, **UNIX Time-Sharing System: Circuit Design Aids**, *Bell System Technical Journal* **57** (6), pp. 2233-2249 (1978).

A. G. Hume, **Mk: A Successor to Make**, *AT&T-BL Computing Science Technical Report 141*, October, 1987.

Mk is available from the UNIX System Toolchest.

# A European Representation for ISO C

*Keld Simonsen*
*University of Copenhagen, Denmark*

*Bjarne Stroustrup*
*AT&T Bell Laboratories, USA.*

Keld is a well known ice cream eating chairman of the Danish group who is active in the Danish group, the network and in the area of international standards.

Bjarne has gained international recognition with his creation of the C++ language. Dr Stroustrup received a MSc in computer science from Cambridge University, England. His research interests include distributed systems, operating systems, simulation, and programming.

Not being native English speakers both of the authors are ideally suited to be heading a campaign to solve the 'European C problem'.

The proposed ANSI C standard uses the American national character set, ASCII, for its representation, but as other national character sets use some characters differently, the ANSI C standard proposal includes a specification for an alternate representation, the *trigraph* representation. This paper proposes an extension to the trigraph proposal based on keywords and two-letter combinations of special characters, that is much easier to read and write. We hope this proposal will stimulate a discussion about this and related problems leading to a widely accepted solution.

## The problems

Writing programs in C can be quite difficult where English is not the native language. The problem is that C uses symbols from the American national character set as operators and punctuation characters. This implies that you cannot use all letters from a national alphabet in identifiers and that programs appear strangely transformed on your screen or printer because some C operators are printed as the corresponding letters.

In many countries in Europe, South America and Africa the national character set standard is a version of the ISO 646-1983 international standard for 7-bit character sets. ASCII (ANSI X3.4-1968) is the national version of ISO 646 in the USA. It is the impression of the authors that ASCII and the national standard competes bravely from country to country for being the national industry standard. Also, on IBM mainframes there are national EBCDIC implementations very close in character repertoire to the formal national standards. In some areas, such as Scandinavia, the national character sets are predominant both in the 7-bit and the EBCDIC world. In other countries, such as the Netherlands, you mostly see ASCII. There are some new international standards based on 8-bit characters (the ISO 8859/1/2 Latin character sets and the ISO 6937/2 Videotex standard) which includes both ASCII and the various national characters, but it will take years, if not decades, for these new official standards to become industry standards and for 7-bit equipment to completely lose its importance.

It is quite desirable (and only polite) to allow programmers to write (and read) programs using their national 7-bit character set representation. The recent proposed ANSI C standard allows a representation of C programs in a national character set that does not have all of the characters used to represent a C program in ASCII. This is achieved by defining alternative representations, trigraphs, for the "offending" C operators and punctuation characters. However, there is still no provision for identifiers with national letters and the trigraph representation is not very suitable for human eyes.

## ASCII and Trigraph Representations

The ISO 646-1983 standard leaves 12 positions undefined and up to national standards organisations to decide. For those undefined positions used in C for national ISO chars, the proposed ANSI C standard specifies a trigraph representation, a three-character replacement beginning with two question marks (??). Three characters ('$', '@' and '`') are not used in the proposed ANSI C (but they are used in some dialects of C) so we will also refrain from using them.

The 12 'problem' characters in ASCII and their equivalent ANSI C trigraph representation are:

| | | | | | |
|---|---|---|---|---|---|
| ASCII | # | $ | @ | [ | \ |
| Trigraph | ??= | | | ??( | ??/ |
| ASCII | ] | ^ | ` | { | \| |
| Trigraph | ??) | ??' | ??< | ??! | ??> |

ISO646 reserves the characters # and $ so that they cannot be used for letters of a national alphabet. This ensures that they are not a problem in the context of C. The other 10 characters are "available for national or application oriented use."

Let us see how these things looks on a common example. First the ordinary C version in plain ASCII:

```
main(argc, argv)
int   argc;
char* argv[];
{
    if (argc<1 ||
            *argv[1]=='\0')
        return;
    printf("Hi,%s\n",argv[1]);
}
```

And this is how it looks in the Danish ISO 646 character set:

```
main(argc, argv)
int   argc;
char* argvÆÅ;
{
    if (argc<1 ØØ
            *argvÆ1Å=='Ø0')
        return;
    printf("Hi,%sØn",argvÆ1Å);
}
```

Experience shows that at least some people can learn to read and write this. In our opinion this is not a skill anyone should be encouraged to enquire. The begin-end brackets and the or operator look ugly and the array subscripts simply drown. It looks just as weird in, say, the Finnish, French, German, or Spanish ISO 646. This is the reason for the proposed ANSI C trigraph solution. How does our example look using trigraphs?

```
main(argc, argv)
int   argc;
char* argv??<??>;
{
    if (argc<1 ??!??!
            *argv??<1??>=='??/0')
        return;
    printf("Hi,%s??/n",argv??<1??>);
}
```

Where are the array subscripts? Which operator is used in the conditional statement? We believe that even though the problem of defining an ISO 646 representation is solved by introducing trigraphs, the original problem of being able to write C programs without losing one's native language has not been attacked at all. The resulting "C programs" are unreadable and unwritable.

We see no alternative to using trigraphs for representing {, }, \, etc., in strings and character constants. For example:

```
switch (tok) {
case '{':   ...
case '}':   ...
case '\\':  ...
}
```

becomes

```
switch (tok) {
case '??<':  ...
case '??>':  ...
case '??/??/':  ...
}
```

Trigraphs are not pretty, but with the notable exception of backslash they will not be common in this context.

## Our Proposal

To solve the original problem one could provide a combination of new keywords and reasonably nice-looking one- or two-letter symbols.

New keywords:

```
or       |
cor      ||   (conditional or)
and      &
cand     &&   (conditional and)
xor      ^
compl    ~    (complement)
```

Introducing new keywords is always a way to break existing programs, but the keywords could be conditionally in effect for new programs. The new keywords requires a few more keystrokes than the ASCII characters, but not many, and some would consider them to improve readability of C! The keywords `and` and `cand` were added for consistency—it seemed silly to have an `or` but no `and`. The `&` and `&&` operators are still valid. The new keywords (except `cor` and `cand`) can be combined with `=` to make assignment operators.

Note that use of trigraphs will not by itself avoid the problem of keywords. Macros will be used to make trigraphs more palatable. For example:

```
#define begin ??<
#define end   ??>
#define or    ??!
#define cor   ??!??!
...
```

Naturally, groups of programmers will agree on standard versions of such macros and naturally not everyone will agree on the same versions. In a few years we will therefore be faced with the problem of having several incompatible sets of "de facto keywords" and conventions. The alternative is to pick a minimal set now. This is what this proposal does.

```
New symbols:

(:              {
:)              }
!(              [
)               ]
??/             \
```

The use of `??/` as the escape character is probably the least elegant of these alternative representations, but we see no acceptable alternative given that trigraphs are necessary in some contexts anyway. One might argue that an escape need not to be too pretty anyway.

We decided to make the compound statement brackets digraphs, finding 'begin' and 'end' too long to write and too likely to be found "not in the spirit of C" by large numbers of programmers. The digraphs `(/ /)` might have been preferable to `(:` and `:)` but current usage precludes this. For example

```
/*
* argument type commented out
* waiting for ANSI C  compiler:
*/

int printf(/* const char*, ... */);
```

Using `(:` will cause a minor problem for C++ parsers because C++ has the prefix scope resolution operator `::`. For example:

```
if (::open("myfile.c",0)==0) ...
```

This problem can be solved either by a bit of lookahead in a C++ lexical analyser or by requiring C++ programmers to use a space before the `::` operator in this relatively rare case.

The grammar of C precludes using `(` and `)` for subscripting (as is done for many other languages). Using keywords to represent `[ ]` would be unacceptably awkward. Using digraphs would be little better but we could not find an acceptable pair. For example `(/ ... /)` suffers from the problem mentioned above, `(! ... !)` collides with the common usage `''if (!p) ...''` and `(* ... *)` collides with the common usage `''if(*p)...''`. Parentheses are typically unnecessary for subscripting and `!` should be considered an infix subscript operator (as in BCPL). The binding strength of the binary `!` operator (subscripting) should be just above the unary operators and it should be left associative. For example, `a!b.c!2*d` means `((a!(b.c))!2)*d`.

Our proposal would allow the program look like this:

```
main(argc, argv)
int   argc;
char* argv!();
(:
    if (argc<1 cor
            *argv!(1)=='??/0')
        return;
        printf("Hi,%s??/n",argv!(1));
    :)
```

Using ! as an infix operator, this can be further cleaned up:

```
    main(argc, argv)
    int   argc;
    char* argv!;
    (:
        if (argc<1 cor
                 *argv!1=='??/0')
            return;
        printf("Hi,%s??/n",argv!1);
    :)
```

One would still need parentheses for more complicated subscripts; for example, `v!(i+j)`. The symmetrical nature of subscripting in C will actually be more obvious. An apparent ambiguity would exist between ! used for subscripts and ! used as the negation operator. Consider:

```
    int v[!4];
```

This would have to be represented as

```
    int v(!(!4);
```

since

```
    int v!!4;
```

would mean

```
    int v[][4];
```

Programs written this way still have a distinct C flavour. Locally, they can even be more terse than C and they are always shorter than programs written using the trigraph notation.

## Practical Details

As observed, the trigraph proposal solves only a small part of the problem they were introduced to compensate for, whereas this proposal is an almost complete solution.

It is debatable whether it should be legal to mix the constructs proposed here, such as `(:`, `or`, and `compl`, with traditional C constructs, such as {, |, and ~.

Mechanical translation from the standard American/English notation for C to this proposed "European" C notation is trivial. The potential ambiguity of translating [ ] into ! is handled by using parentheses systematically; `[expr]` becomes `!(expr)`. The reverse translation requires understanding of operator precedence to handle subscripting. Furthermore, if national characters are allowed in identifiers such characters must be expanded into sequences of English characters. This is no major problem.

## Conclusions

We have found a *usable* solution to the international representation problem of C for most languages using a latin alphabet. The solution is quite easy to implement in a C compiler. We tried it.

The `(:`, `:)`, and ! constructs do not affect existing programs, and the new keywords are chosen so that it is likely that only a minimal number of existing programs will be affected. Using them only in programs identified as non-US ISO C should remove completely the possibility of hurting existing programs.

If adopted this proposal will open the way for allowing national characters to be used in identifiers.

This proposal is submitted for consideration to the ANSI C committee and the ISO committees dealing with C. It is also considered for adoption into C++.

## Acknowledgements

Brian Kernighan and David Prosser found errors in previous versions of this proposal and suggested improvements to the presentation.

## References

K&R: C

ANSI C draft

Richards: BCPL

BS: C++

```
Received: from roof.gardenshed.jocks-cabin by patio.macmiggins.stornoway via
        shoutnet; 21 Jan 89 20:50 SST
Received: from patio.macmiggins.stornoway by bonfire.oban.beaconnet via
        infernonet; 25 Feb 89 15:31 SST
Received: from bonfire.oban.beaconnet by cs.nott.ac.uk via
        cannet; 30 Mar 89 21:43 BST
To: jpo@cs.nott.ac.uk
From: jock@gardenshed.jocks-cabin
Date: Fri, 21 Jan 89 20:50 SST
Subject: Alternative protocols
```

# Alternatives to OSI

*Jock C St. Martin*

*University of the Outer Hebrides*
*Scotland*

Following recent discussions concerning the relative merits of OSI and ARPA protocols, I decided to throw my hat into the ring. Furthermore, I believe that the ARPA protocols are not the only contenders with OSI, and that a number of even more 'mature' mechanisms exist. I present seven possibilities for consideration.

## 1. Bean tins and bits of string

The use of bean tins and taut pieces of string has long been recognised as an effective means of communication. In fact, excavations from Anglo-Saxon dwellings in Nottingham show their use (albeit with imported coconuts as opposed to bean tins) in early everyday office situations.

Bean tins and string have several advantages over OSI:

a. They are fast, light weight and portable.

b. They don't require the purchase of expensive computers.

c. Complex error correction (based on the ''NO - I said ...'' principal)

d. Uses off the supermarket shelf technology.

e. They were not invented by the ISO.

They also exhibit a very few trifling limitations:

a. Poor support for 'packet' switching (however, tin switching may be supported).

b. Users often cut themselves on the tins.

c. Star network topologies become more complex.

d. They don't scale very well.

## 2. Shouting from the roof tops

Shouting from the rooftops can be an effective method of optimised local area communication. It is based on the well understood CMSA/CD technology but with the notion of priority. Users can insert high priority traffic with the ''If I might get a word in edgeways'' packet. It is already in widespread use—e.g., the House of Commons, political canvassing and Speakers Corner. Naturally, a roof top is only necessary for high bandwidth traffic. The PTT's would probably assume this role. The average user would be content to shout in the street.

Shouting has many advantages over OSI:

a. It is not as 'complex and obscure'.

b. Most people understand shouting.

c. Broadcasts are easy.

d. It's fun.

e. It wasn't invented by the ISO

OSI has hardly any advantages over shouting:

## 3. Burning beacons on hill-tops:

Burning beacons on hill-tops have long been used to warn of advancing Armadas and their like. However, the author believes that beacons may have wider applications than just these.

In particular, they have the following advantages over OSI:

a. No 'dangerous checkpointing'.

b. They keep you warm.

c. Not overly complex and obscure.

d. A secondary use for the disposal of those nasty ISO people.

e. Not cluttered with unnecessary functionality.

f. Not invented by the ISO.

Disadvantages to OSI:

a. Not suitable for the office environment (this may really be an advantage in some circumstances).

b. Low bandwidth (may also be an advantage—see 7)

c. Error rates can be high. Arsonists, pyromaniacs and 'Satanic Verses' burners can generate spoof packets.

## 4. Semaphore

Semaphore has been in use for many years. So why did ISO not consider this for international internetworking? This is difficult to determine, but is probably due to political motivations rather than any deficiencies in the protocols. Naturally there are a few rough edges to be addressed.

Advantages over OSI

a. Broadcasts are easily accommodated.

b. Widely supported off-the-shelf infra-structure (boy scouts).

c. Not invented by ISO

Disadvantages over OSI

a. Not so useful at night (but a working party on luminous flags is in progress).

b. Bandwidth is rather low—but automation should help.

## 5. Messages in bottles

This is a low cost solution to networking. Bottles are easy to obtain and with a little development, this neglected backwater of communications technology could be a real alternative.

Advantages over OSI

a. High bandwidth data channels already in existence (e.g. the gulf stream, rivers and sewers.)

b. Large amounts of data can be placed in the appropriate sized bottles.

c. Not invented by ISO.

Disadvantages to OSI

a. Transit time is unpredictable (but then IP, for instance, does not guarantee any bounded delivery time)

## 6. The Telephone

This might be seen as an enhancement of method 2. However, there is a lot to be gained from this approach. The name lookup problem is already solved as are routing issues. Lets face it, communications protocols are ultimately used for communicating between people. So why not just standardise the telephone. Add on services such as broadcast agents (commonly called gossips/operators) are easy to achieve.

Advantages over OSI

a. Its a mature existing technology.

b. Directory services issues, routing and charging are already established.

c. It's now available in portable form.

d. Not invented by ISO

Disadvantages to OSI

a. Because it's a mature technology, there aren't so many interesting research areas.

b. As a result of 2. there are few exotic conference openings. It costs money.

## 7. Not communicating at all

One question I asked myself was "why communicate at all?" On consideration it was realised that not communicating has the following advantages over OSI.

a. Low consumption of bandwidth.

b. Cheap and easy to manage.

c. No one disagrees with you.

d. Without the time wasted on communication, other business proceeds much quicker.

e. Not invented by the ISO

No known disadvantages to OSI.

## The ARPA protocols.

The ARPA protocols deserve consideration along with many of the above mentioned methods of communication. In particular, they have one major advantage over OSI.

a. Not invented by the ISO

However, despite this overwhelming advantage of the Internet protocol suite, the ISO proponents simply will not give in. In this section I therefore give a few other reasons for the superiority of the Internet suite—as if 1. was not enough.

Scalability. The Internet protocols are obviously scalable as has been proved time and time again. All that is required is for the PTT's to take the sensible step of providing a network infrastructure and the rest can be solved. Charging is easily accommodated—the PTT's pick up the bills.

Network interface. Many people have commented on how convenient it is to have a network address which fits into a common word size. This is such a advantage that the limitations are really insignificant. If the address space ever gets used up there is an obvious extension mechanism—the waiting list.

Session layer. The Internet suite sensibly disregarded session services as superfluous. As has been observed, checkpointing is inherently dangerous as it can lead to loss of network usage and revenue. OSI has been influenced by the Internet community here, and has provided a session service complex enough that most implementations try and ignore it.

Presentation layer. Again the Internet triumphs. It is quite clear that for the most part applications only need to exchange data consisting of bytes of 8, 16 and 32 bit quantities. These simple structures can be used as building blocks to construct almost any structure required. If this is not sufficient, there is a simple escape mechanism provided, known in the jargon as a ''string encoding''. It is quite clear that ASN.1 is just over the top—CHOICE's and OPTIONAL's are for quiche-eating indecisive applications.

Application layer. Well the Internet has got this one too. Honestly, it's quite obvious that each application should do its own thing. That's what they're there for. If an application needs remote procedure call interface, or security, or name lookup, then it can do it itself rather than forcing it to use some more general service like ROS or directory services.

## Summary

In summary, I feel that all of the above methods are orders of magnitude better than OSI (which incidentally, and by coincidence, wasn't invented here). In particular, I feel that method 7 offers the greatest potential and, with this in mind, WE DO NOT WELCOME ANY FURTHER COMMENTS YOU MIGHT HAVE!

### Editor's note

This article is in no way connected with either Julian Onions or Steve Benford of the University of Nottingham beyond their rôle as postal agents for the author.

# Brussels conference report

*Marc Gumbold*
*marc@pbinfo.uucp*

*Universität-GH Paderborn*
*West Germany*

Marc Gumbold is a graduate CS student at the University of Paderborn, West Germany.

He finds it rather hard to survive without money and is therefore spending some of his time as *postmaster@pbinfo.uucp*.

Although he has not yet been to Italy he is really fond of Cappuccino and Espresso, but doesn't disdain other kinds of coffee either.

Once upon a time there was a German student who decided to go to an EUUG conference for the very first time in his life. He sent email off to *euug@inset.uucp*, which he found to be the uucp address of the EUUG Secretary, in order to get more information about the Spring '89 Conference. Some 10 or 12 days later he eventually received a nice booklet containing the Conference Programme and—shock!—the costs. As he would like to attend the *UUCP & Sendmail Configuration* tutorial very much, he would have to pay £475 altogether. Impossible! Rather annoyed he read the chapter about 'Student Grants' and wondered whether a student grant application would be accepted. As there were only a few days remaining before the booking deadline he eventually decided to make a phone call to the EUUG Secretary in England. A kind gentleman (who you might guess is Bill Barrett) on the other end of the line told him that they will pay the fee for the Conference and one tutorial for him, if he just agrees to write something for the EUUG Newsletter. Of course he accepted.

Well, this tale is actually the prehistory of (a) my participation in the EUUG Spring '89 Conference in Brussels and (b) this report. You already guessed *that*? Ok. Never mind. Let's start the action.

Considering my financial situation I tried to book a bed in a youth hostel. I got a bed for all but the first night, but wasn't disturbed too much by this as I thought there would be no problem in finding some place to sleep the first night, once I was in Brussels. As Brussels is really not too far away from Paderborn (perhaps some 500 kilometers) I decided to go there in my own 12-year-old veteran car.

## Monday

For various reasons I couldn't start on Sunday so I left Paderborn *very* early on Monday morning heading for Brussels. Not being too much troubled by the snow on my way I approached Brussels at about half past seven in the morning. I got stuck in the morning rush hour. It was eight o'clock when I found myself in the very center of Brussels. Imagine the situation as follows: Holding the city map in the one hand, turning the steering wheel with the other, and changing gears with the third I was looking with one eye on the city map, with the other on the street to watch traffic, and with the third I was looking out for the Palais de Congrès where the Conference was to be held. (You will easily notice a slight mistake I made; there are a limited number of resources like hands and eyes of course. However I would not like to explain the scheduling algorithm to you right now...) Perhaps this would have ended up in chaos if not my car started making some strange noises. So I had to stop. A quick investigation revealed the cooling water had boiled, so I had enough time to ask an elderly woman who just came along, where the Palais de Congrès could be found. She spoke French :-(, found out I was German :-), and continued to speak French :-(. Well, nevertheless somehow I understood her. Actually the Palais was only one or two hundred meters away. I managed to get my car in an underground car park near the Palais and went off.

The Palais de Congrès is situated on the *Mont des Arts* or *Kunstberg*. Both means *Mountain of Art*. This is a really funny thing in Brussels: all the streets and places are labeled in French and Dutch. You'll perhaps know that people in northern Belgium speak Dutch (well, really Flemish, but don't be fussy about that, it's roughly the same), and those in southern Belgium speak French. But although Brussels is located north of the language border, most of the inhabitants speak French.

Well, let's return to the Palais de Congrès. Entering an entrance labelled 'EUUG' I went straight to the registration desk. I was welcomed by Bill Barrett. He immediately remembered my name and handed over the Conference Proceedings and my name badge to me.

There was still some time remaining until the beginning of the Conference. I had a look around. The conference room itself was rather big and equipped with nice and comfortable seats in red and orange tones. After a 'phone call home I decided to sit down in the foyer and wait for things to happen. More and more people dropped in, got their proceedings and their badge and did the same I was doing. I met a couple of people from the German EUnet backbone in Dortmund who I had seen before. Sigh! At least someone I knew a little.

Well, eventually we sat down in those cosy seats, waiting for the Conference Chairman, Prof. Marc Nyssen from the Free University Brussels, to open the Conference. He found some nice words to welcome us and presented a keynote speaker, Dennis Tsichritzis. Dennis spoke about the need for object oriented program development. I really got the feeling that what he said was rather interesting but his speed and non-standard English made it rather hard for me to follow.

With some other 400 people I rushed out to get coffee. There was a slight problem with my stomach as it thought it was lunch time already (remember I got up at 2.30 in the morning). Walking around I got the impression that everybody else seemed to know everybody except me. Hmmm. But I felt somehow attracted by the multi-lingual noise level surrounding me, having a glimpse at all the badges showing where people came from. Denmark, France, UK, Germany, Netherlands, even people from the States.

The next session was dealt with distributed operating systems. It began to get really boring. I leaned back and wondered if it had been a really good decision to go to this Conference. But the last talk rather changed my mind. Sape J. Mullender reported about their Amoeba system development in a very impressive manner. I even forgot to think about my rumbling stomach.

Finally, lunch. There was a restaurant inside the congress center where lunch was served. During the meal I took a quick course in Belgian history, given to me by a Michel Bardiaux from Brussels, who was sitting next to me.

The next three talks were entitled 'Networks'. First, John S. Quarterman, American network guru, reported about 'Recent changes in North American Computer Networks'. I was really confused by the amazing number of networks they have over there. If you nevertheless would like to learn more about this: John will publish a book about these topics in this year. After a contribution from Austria about an educational network, Daniel Karrenberg gave an overview about EUnet, the European UNIX Network. He also presented the teams managing the national EUnet backbone computers. A very nice idea, every team showed some slides about what their site looked like.

Coffee again (it was really no problem for coffeeholics like me to survive this conference...). Chatting with the Dortmund backbone gurus revealed that there was one empty bed in one of their hotel rooms. So at least I knew where to spend the night.

The next and last chapter for this day was 'Graphics', including two very interesting items: First the development of an medical image processing system on a SUN workstation, and second a five minute computer animation which was computed in a network of SUNs using only the machines' idle times. This one was done at the University of Karlsruhe. Really impressive!

After having some free drinks (just orange juice for me, as I had to get my car out of the car park), me and Jan-Hinrich Fessel from Dortmund (the guy who agreed to share his room with me for one night—thanks again) went to the 'Hotel Madeleine'.

After relaxing for a short moment we decided to find some place to eat something. As there were six of us, everybody having different criteria of choosing the right pub, it took us a rather long time and some kilometers of running around, until we all were hungry enough to agree to enter one pub near the *Bourse* (or *Beurs* which all means *stock exchange*). We had some beer and something to eat, which was really not as expensive as you could think, considering the nice interior decoration which probably had not changed much since the early 20ies. I'm afraid I must disappoint you in not being able to give you a report on the quality of Belgian beer (I think you should ask *postmaster@unido.uucp* for that...). I'm in no way a beer professional, I only drink it in case of extreme thirst. If you would like to chat about coffee, you're welcome.

Now I became really tired, after all I have been on my legs for more than 20 hours. After having a look at one or two more pubs I decided to return to the hotel and have some sleep( ). But the hotel was in the very center of the old town, including masses of cars driving around all the night over streets with cobbled pavement. However I managed to sleep a few hours.

## Tuesday

After a lousy breakfast in the hotel I transferred my car and my luggage to the youth hostel. I found a car park *free of charge* near the youth hostel. Now that was something! I had paid 320 Belgian Francs for car parks since my arrival on Monday morning. (Look it up in your newspaper if you would like to know how much that is...)

From the youth hostel I reached the Palais de Congrès with a five minute walk. It was an ugly cold morning and I was really sleepy. Nevertheless I rather enjoyed the first talks about 'Network Software'.

Coffee break. Outside it was snowing now. Brrrr...

Luckily there were no windows in the conference room itself, so you could at least imagine that the sun was shining outside. Well, inside we were listening to the very interesting talk of David Rosenthal from Sun Microsystems. He spoke about the relationship between response times and memory, but it was much more interesting than it probably sounds. I really encourage you to read his paper 'More Haste, Less Speed' in the proceedings, especially if you're a workstation user.

Lunch. While I find it a bit difficult to get in contact with others during the coffee breaks, it's much easier during lunch. This time I had a chat with Claude Anzala, from the French EUnet backbone, about some mail problems. It was especially nice to speak to people you only (so far) knew by name; Teus Hagen for example.

The whole Tuesday afternoon was about 'Standards'. I was especially looking forward to a talk named 'UNIX Standardisation: an Overview', but was a bit disappointed that this talk was rather a description of the POSIX concept than an overview over existing standards. Next Robert L. Duncanson from AT&T Europe reported about System V Release 4, followed by Kevin R. Fall from Berkeley speaking about new features of BSD UNIX 4.4(?).

Coffee (sigh!) was followed by John Totman's talk about X/Open. A change in the programme and the next talk was held by Morris Schwartz from 'UNIX International'. Phew! I've heard about X/Open, OSF, POSIX, and so on, and I must say I was already confused by all these standardisation efforts. But now still another group, which was presented to be the 'System V Product Planning Authority' and which will support X/Open. I really began to loose the thread totally. Then Henning Oldenburg from OSF who explained some things about OSF/Motif, yet another window system standard... oh no!

I was tired. And I was really fed up with all this standardisation stuff. There was a BOF (Bird Of a Feather meeting) about standards, but I decided to have a look at my youth hostel to relax a bit. Besides there was the Conference Dinner at eight o'clock in the 'Autoworld Museum'. Laying on my bed I tried to work out how to get there. For several reasons I didn't want to go there by my own car, a taxi was considered to be too expensive, it was too far to walk, so I decided to go by bus. Line 20 would be ok, there was a bus stop very near the youth hostel, so I would have no problems.

A little after half past seven I boarded a bus. It was a wonderful tour; the bus was a real veteran, the streets were bumpy, so I soon felt every bone inside me. Having the city map on my knees I tried not to loose orientation in order to get out at the right bus stop. I noticed another passenger who was equipped with a city map. He got out at the same bus stop and followed me. I was not surprised to find out that he planned to go to the EUUG Dinner. After a few minutes we luckily found the entrance to the Autoworld Museum.

Really amazing, I had never seen such a great number of old cars before. I got a drink, talked to some fellow EUUGians and admired all these lovely old cars. They were even older than mine! I was especially attracted by a bus, who seemed to be in a much better condition than that one that took me there and I really wondered why this one was staying in a museum considering the much worse material on the streets outside.

Finally, the dinner. There was nothing extraordinary about it, except for the fact that, in contrary to the lunches, people had to get up from their seats, and go to one of three (or four, I don't know exactly) tables to get the food onto their plates. Of course this led to race conditions...

Now I really had a problem: The youth hostel closed at midnight, I didn't know how to go back (bus lines seem to be unidirectional in Brussels) so I had to get up earlier than the rest to reach the hostel in time. (I always had the dark feeling there was a hitch to the low price of a youth hostel...) However I managed to get back by metro and reached the youth hostel at five minutes to twelve.

## Wednesday

The first talks on Wednesday morning revealed nothing special. No, the talk of one Japanese (!) should really be mentioned. He spoke about the development of a high speed filesystem which becomes necessary if huge masses of data, for instance video data, must be accessed.

Tom Killian from the AT&T Bell Labs presented an improved 'make'[1], called 'mk'. In addition to the well-known 'make' it understands meta-rules defined by regular expressions and transitive closure of dependencies.

So far I have forgotten to mention that there was a competition. The task was to produce net addresses that were likely to me more descriptive than those normally used. After lunch on Wednesday the results were proclaimed:

The winner was Dominic Dunlop with *worm@broadcast* for which he was rewarded with a UNIX licence plate.

The second prize, a T Shirt was given to Paul Bussé for:

*five%work@myhost*.

Some other results listed for your enjoyment:
Honourable Mentions:

*i-am@cinema.darling*, Eva Kühn.
$+\<@$-1\>$*, Philippe Dax.
hacker@*, Peter Gilis.

Nepotism section:

*tilbrook@another.new.job*
*sunil!@home*
*keld@nearest.icecream.stand*
*red-shoe@michel.right-foot*

Novel Transport layer:

*?#^.?@toilet.paper*,
/* the address was actually written on toilet tissue ... */

In the afternoon we learned something about languages. Two object-oriented talks, and one about distributed logic programming. It seems to be quite fashionable nowadays to distribute all sorts of thing that used to be sequential... The last group of talks on Wednesday afternoon was about 'Tools', but I skipped most of them because I had a look at the exhibition in the basement... Oh yes, the exhibition. I almost forgot it. You could find many really great names there, but I got the impression that there was rather little interest in the exhibition at all. At least until they transferred the coffee break facilities into the basement room where the exhibition took place. Then suddenly the room was overcrowded. Coffee speaks louder than sales talks...

One Belgian firm however managed to present Steve Jobs' NeXT computer. There was always a reasonable crowd around the black cube. I would really love to have such a box, especially for writing this report; that thing's got a built-in dictionary and thesaurus... however when testing it I could find no facility for providing you automatically with ideas as to what to write about... Alas, no machine is perfect.

The meaning of the first letter of 'EUUG' was impressively demonstrated by the fact that one talk was given by Agnes Hernadi, coming from the Hungarian Academy of Sciences in Budapest. She told us about the development of a new, unusual database interface. The very last talk was given by Lee McLoughlin. Lee spoke about his experiences with porting X Windows to different machines, giving hints for all the poor chaps who have to do the same.

1. Tom's paper can be found elsewhere in this newsletter.

Well, Wednesday evening was the official ending of the Conference. Teus Hagen and Marc Nyssen found some nice words—all of which I have forgotten. In the evening there was a EUnet BOF, where several problems concerning the EUnet situation were discussed.

Although the weather was terrible (what luck! I found an umbrella in my car) I had a walk through the old center of Brussels in the evening. I walked over the Grand Place with the famous town hall, but because of rain and coldness nothing could impress me. So after a pizza and an espresso I ended up in the youth hostel, talking the rest of the evening with *normal* people, i.e. those that were in no way related to UNIX. After 3 days UNIX this can really mean some relief.

## Thursday

As I planned to go to the *UUCP & Sendmail Configuration* tutorial on Friday I had to stay in Brussels. I could really have gone home now, with all these new impressions I had. Besides, the weather... you already know about. But as I was looking forward for the tutorial I decided to make Thursday my sightseeing day. So I spent three quarters of the day running around in the city. I was really impressed by the amazing number of restaurants and pubs in the old city. There are lanes with, let's say ten or more pizzerias; if you're running out of one you'll find yourself in the kitchen of the next one. Well, I had a look at the 'Manneken Pis', of course. I heard before that it sometimes wears clothes, but was nevertheless impressed finding it dressed with an MP uniform. Although this was surely a respectful appearance I couldn't hide a smile... By the way, do you know there is a feminine counterpart to 'Manneken Pis'? It's called 'Janneke Pis' and was made a few years ago. Just drop an email if you would like to know where to find it. Running around I really got the impression that Brussels was rather a French town, not as clean and tidy as Dutch or Flemish towns are. But I'm not sure if I would have got another impression if there would have been a little bit sunshine. The weather was a real misery. After some shopping I returned to the youth hostel in late afternoon, wet to the skin. I warmed up myself. Because of the fact that there was a small party in the youth hostel, the following night consisted only of some 3 hours of sleep.

## Friday

Sleeeeeeepy. How could I manage the tutorial? Well, somehow I got to the Palais de Congrès and found out that the tutorial started at 9.30. Hmmm, one should really read the timetable. What a pity! I could have had another half an hour inside my sleeping bag.

I checked in for the tutorial and received the tutorial papers. They looked rather promising. The tutorial dealt with the configuration of *sendmail*, a sophisticated mail transport agent, and was given by Yves Devillers, mail guru at the French EUnet backbone. You probably either know by experience or have heard from others that *sendmail.cf*, the sendmail configuration file, causes nightmares for most postmasters. Well, it surely didn't do so to Yves, who managed to tame the beast. The mystery of his success was sendmail's debug option, which he tried to explain to us. The pace was all right, although there's much more to say about sendmail than could be done in one day. But the provided papers were really excellent, so I think it's no problem to make more use of this tutorial working on the papers at home.

At five o'clock in the afternoon it was all over. I got out of the Palais and found the sun shining for the first time of my stay in Brussels. I just couldn't believe it, after 4 days of rain and snow. I returned to the youth hostel and gathered my luggage. Fortunately I had not to travel alone. I was accompanied by Anke and Bernard from the German EUnet backbone, and Kevin Fall from Berkeley, who decided to go to Germany. I dropped them in Dortmund and arrived at home happily but absolutely tired at exactly midnight.

The conclusion? This was my first EUUG conference but *surely* not the last one. I really enjoyed it. You'll probably not find me in Vienna, because of an exam I've to do in late September. But I'll certainly meet you in Munich next year. I would like to encourage every student who feels somehow related to UNIX, to have a try and go to an EUUG conference. Costs are a problem for students, of course. (Not every chap can pay for the conference by writing a report—who would care to read all these boring essays? :-)).

See you in Munich, Marc.

# News from the Netherlands

*Frances Brazier*
*frances@psy.vu.nl*

*Department of Cognitive Psychology*
*Vrije Universiteit*
*Amsterdam*
*The Netherlands*

Frances has a master's degree in Mathematics and Computer Science, and has been doing research at the Department of Cognitive Psychology for the past 7 years. Human-machine interfaces and information retrieval are her major fields of interest.

## Membership

The NLUUG's membership continues to increase in number. In the table below the increase in academic, industrial and individual membership is shown.

| April 88 | Nov 1988 | April 89 | |
|---|---|---|---|
| 59 | 71 | 80 | academic members (@ Dfl 300) |
| 133 | 157 | 164 | industrial members (@ Dfl 600) |
| 20 | 31 | 35 | individual members (@ Dfl 100) |
| 212 | 259 | 279 | TOTAL |

## Conferences

Our next conference will have been held by the time this newsletter is printed—on May 9th. The theme of this conference is human-machine interaction, a topic we expect to be well received. We have again invited a few 'foreign' speakers to broaden the scope of the programme, one of whom previously spoke on the same topic in Lissabon. The exchange of ideas and of speakers between EUUG conferences and national conferences is one which should be promoted. As has been shown in the past, both parties can profit from each other's efforts. Good speakers aren't always easy to find, let alone good talks.

Our first experience with the production of proceedings was gained last autumn. Although the proceedings were well appreciated it was decided to leave the decision to produce proceedings up to the programme committee. This has resulted in the decision to defer from the production of proceedings for the May conference. The general aim is, however, that proceedings should be published for at least one of the two conferences held in a year. Speakers will be informed of the possibility to submit their contributions to this newsletter.

The programme for May 9th includes:

- *Natural Language Communication with Computers: Some compromises and problems.* A Jameson & W Claassen. Nijmegen Institute for Cognitive Research and Information Technology.
- *The Océ 6000 publishing and printing system: the development of a graphical user interface.* René Collard. Océ Nederland BV.
- *Windows.* Shawn Phillips. SUN Microsystems Inc, Mountain View, USA.
- *NeWS and X, Beauty and the Beast?* WT Roberts. Department of Computer Science, Queen Mary College, London.
- *The Generation of Graphical User Interfaces using Attribute Grammars.* S Doaitse Swierstra, HH Vogt, H Bos. State Department of Computer Science, University of Utrecht & HCS Information Technology.
- *An object-oriented approach to UI-design.* Charles van der Mast & JM Versendaal. Department of Computer Science, Technical University of Delft.
- *X-Window and Open Dialogue: a standard UIMS.* Bouwe van der Eems. Apollo Computer BV.

## The backbone

A new legal body has been established for the exploitation of the new Dutch backbone *hp4nl*: Stichting NLnet. The Board of this Foundation consists of 3 to 5 members of which the majority must legally be appointed by the NLUUG. Currently the Chairman, Treasurer and Secretary fulfill the same positions for both the NLUUG and the Stichting NLnet. A candidate for system management has been found. If everything goes well he will be stationed at CWI as of May 1st.

As of May 1st our network accounting has also been revised—the principle 'sender pays' has been implemented. We now just hope the algorithm's correct. All mail is classified as being either national, European or non-European, and is accounted as such.

The problems with SURF (EARN) have not yet been solved completely. We do, however, hope to have a contract signed by the beginning of May.

# EUUG Executive Report

*Helen Gibbons*

Helen Gibbons is also the business manager of the EUUG and is contactable at the EUUG secretariat.

The EUUG Executive Committee held its last meeting on 1st April 1989 just prior to the Spring Conference in Brussels. This was followed a day later by the Governing Board Meeting at which national group representatives from Denmark, Sweden, Italy, Germany, The Netherlands, Portugal, Brussels, Ireland, Norway, France, Yugoslavia, Austria, Great Britain and Finland were represented. There was also an invited observer from USENIX.

Earlier in the year an application for affiliation from Hungary had been accepted, and at the April meetings official approval was granted also to applications recently received from Portugal and Yugoslavia. This means that there are now 16 National Groups within EUUG with a growing membership totaling around 3000.

The Swiss group has been suspended from membership.

The year 1988-89 saw a big improvement in EUUG services, particularly the EUUG Newsletter which grew to be bigger and better than ever before; other publications were also made available, such as the BSD Manuals and E-mail directory, which was a big success and is selling well. Conferences, tutorials, software distributions, public relations and EUnet are all becoming bigger, busier and more professional. This, of course, means not only that investments and budget risks are higher but the work load has also greatly increased. Because of this, the position of an EUUG Executive Director is being considered. The Executive Committee itself will also be expanded to cope with the work load. It will meet again early in July.

The EUUG has been active in its contact with other bodies and groups. Discussions with NCR and SUN user groups are projected, but have not been very successful so far. An exchange of articles has been arranged with USENIX and co-operation is also taking place on the subject of standards. Meetings are being arranged with OSF, UNIX International and X/Open.

EUnet is moving steadily towards a stable and very reliable service. Most of the core links use either leased lines or X.25. The financial situation of EUnet is now improving and it is expected that the US sites will soon contribute to the US European data transportation cost. It is hoped that the 'mcvax' machine will shortly be replaced by a SUN 4 server. Thanks are due to the co-operation of the many network managers at all the European backbones.

The future strategy of the EUUG will be discussed in great detail at a Strategy Workshop being held in Dublin on 5-7th May 1989.

Although some discussion was held at the April meetings on reducing the number of major conferences to only one a year, it was in the end agreed that the EUUG would continue with its programme of two conferences per year for the forseeable future. The Spring Conference in Brussels was a great success with an attendance of approximately 400. Future conferences will be held in:

| | |
|---|---|
| Vienna, Austria | 18-22 September 1989 |
| Munich, W. Germany | 23-27 April 1990 |
| Nice, France | 22-26 October 1990 |
| Tromso, Norway | 20-24 May 1991 |
| Budapest, Hungary | 16-20 September 1991 |

There after conferences are in the early stages of planning for Jersey and the Netherlands in 1992.

EUUG

European UNIX® systems User Group

# EUUG CONFERENCES — The World of UNIX at Your Feet

Twice each year — in the Spring and Autumn — the EUUG holds major International Conferences encompassing all the most interesting developments and activities associated with UNIX.

These events are unequalled anywhere in the world for their content and the very high level of speakers — invited from the leading academic and industrial UNIX centres in the USA, Europe and the Far East.

The importance of the Conferences — which are accompanied by tutorials and exhibitions — is underlined by the fact that well over 2000 delegates have attended the last seven events in Florence, Manchester, Finland, Dublin, London, Portugal and Brussels.

If you are at all involved in the field of UNIX, then the EUUG Conferences should not be missed. Indeed, you can make no better start than to put these dates in your diary:

| | |
|---|---|
| **Vienna, Austria** | **Autumn 1989 (18-22 Sept)** |
| **Munich, Germany** | **Spring 1990 (23-27 April)** |
| **Nice, France** | **Autumn 1990 (22-26 Oct)** |
| **Norway** | **Spring 1991 (20-24 May)** |
| **Hungary** | **Autumn 1991 (16-20 Sept)** |
| **Jersey** | **Spring 1992 (Dates to be announced)** |

or call the EUUG Secretariat for further details:

**European UNIX® systems User Group**
Owles Hall, Buntingford, Hertfordshire SG9 9PL, UK
Tel: + 44 763 73039 Fax: + 44 763 73255
**Network address: euug@inset.uucp**

EUUG

European UNIX® systems User Group

# EUUG Autumn '89 Conference and Exhibition

## 18-22 September 1989

## Wirtschaftsuniversität – Vienna

### Programme

## TUTORIALS

### Monday 18th September

| | |
|---|---|
| Tutorial M1 | UNIX Network Programming<br>Tutor: Richard Stevens, Health Systems International |
| Tutorial M2 | RISC<br>Tutors: Jean Wood and Ashis Khan, MIPS Inc. |
| Tutorial M3<br>(Morning only) | System V IPC<br>Tutor: George Philippot, NCR Norway |
| Tutorial M4<br>(Afternoon only) | MAKE<br>Tutor: George Philippot, NCR Norway |

### Tuesday 19th September

| | |
|---|---|
| Tutorial T1 | Beyond 4.3 BSD<br>Tutors: Mike Karels and Kirk McKusick |
| Tutorial T2 | OSI<br>Tutor: Colin I. Anson, Hewlett Packard |
| Tutorial T3 | The Andrew Toolkit<br>Tutor: Nathaniel Borenstein, ITC CMU |

## TECHNICAL PROGRAMME

The technical programme will commence with an opening session at 09.30 on Wednesday 20th September and will continue until approximately 17.30 on Friday 22nd September. Full details of the technical programme will be published in the conference registration booklet. The subject areas will include Security, Fault Tolerance, Transaction Processing, RISC Architectures, Scheduling in multi-processor systems, User Interfaces, Network Management, Interoperability with other systems, Artificial intelligence and 5th GL, and Standardisation.

## THE EXHIBITION

The exhibition will take place at the Wirtschaftsuniversität on 20-21-22nd September. Companies may choose either an information only or a full exhibition package with their name listed in the conference registration booklet which will be published in May.

Companies wishing to book exhibition space should contact the exhibition organiser:

**Mrs. B. Hainschink,**
**ÖCG**
**Österreichische Computer Gesellschaft**
**Wollzeile 1-3**
**A-1040 WIEN**
**Austria**
**Tel: +43 222 512 0235**
**Fax: +43 222 513 7735**

Stand costs will vary from OS10,800 (£484.74) for members leaflet distribution only to OS36,000 (£1,615.80) for non members for the full exhibition package.

## STUDENT GRANTS

Grants are being offered to assist students to attend the Conference. An application must be made well in advance of the Conference. A decision will be made before the event whether an application qualifies for a grant. Payment will not be made until after the Conference but the applicant will be able to proceed in the knowledge that the grant will be forthcoming.

Priority will be given to:

1. Students giving a talk at the Conference.
2. Students doing work for the EUUG or a National Group
3. Students
4. Other deserving cases like research students.

## HOW TO BOOK

A Conference Registration Booklet giving full details of the programme and events, costs, hotel booking, sightseeing programme, plus booking forms will be printed and distributed at the end of May.

Costs will be as follows:

| | | | |
|---|---|---|---|
| 1 Day Tutorial | 1 person if booked before 17th July | Members Only | £180 |
| 1 Day Tutorial | 1 person if booked before 21st August | Members Only | £195 |
| 1 Day Tutorial | 1 person if booked after 20th August | Members Only | £220 |
| Registration and Payment on the door | | Members Only | £240 |
| 3 Day Conference if booked before | 17th July | Members | £195 |
| | | Non Members | £270 |
| 3 Day Conference if booked before | 21st August | Members | £240 |
| | | Non Members | £320 |
| 3 Day Conference if booked after | 20th August | Members | £270 |
| | | Non Members | £350 |
| Registration and Payment on the door | | Members | £300 |
| | | Non Members | £380 |

## ELECTRONIC MAIL AT THE CONFERENCE

Delegates can be reached during the Conference by EUnet mail. Messages will be printed, sealed and posted to the message board.
To reach people at the Conference use the following address:

firstname_lastname_organisation@euug-conf.eu.net
or
firstname_lastname_organisation@euug-conf.uucp

Delegates should limit the amount of messages forwarded to this address.
Distribution lists should not be forwarded.
Delegates will also be able to send mail.

## CONFERENCE & TUTORIAL ENQUIRIES

All enquiries and requests for the Conference Registration Booklet should be sent to:

**Mrs. Helen Gibbons**
**The Secretariat,**
**EUUG,**
**Owles Hall,**
**BUNTINGFORD,**
**Herts. SG9 9PL,**
**UK**

**Tel: +44 763 73039**

**Fax: +44 763 73255**

**euug@inset.uucp**

# Report from the Danish Group

*Keld-Jorn Simonsen*
*keld@dkuug.dk*

*University of Copenhagen*
*Denmark*

Keld is the chairman of the Danish group and he occasionally writes an article or two for the EUUG Newsletter, where he is also known as one of the 3 great pinup danes from Owles Hall. Currently he is much engaged in the Danish group, the network, and international standards.

The DKUUG had on the 1st April1989 300 members, of which 47 were individual 244 were organisational, and 9 were in a new category called member at large, for big firms. Individuals pay 600 DKK a year, organisational pay 1800 DKK, and members at large pay 3600 DKK.

We have done something to our newsletter, which now comes out about once a month, in a new format. We have employed two people to do the editorial job and also write some articles themselves. The newsletter has become more professional, with almost all articles written in Danish. We also have new printer and distribution arrangements.

On the meetings side we have improved the service by going from four meetings a year to eight. Half of the meetings are in the afternoon, free of charge, the rest are payable. We recently had meetings on graphical interfaces, CASE, Object Oriented Programming, and Office Automation. The next one is a two-day event on networks and security. Meetings draw from 40 to 100 people.

The network is on the upswing too. We have just recently begun employing leased lines to mcvax and the USA, and as a result we now enjoy a much better response time. We have set up a name server and we have had to employ another person half time for 3 months to help with the increased workload. The pricing on the net has changed to a fixed price per year. We currently have 80 on mail and 25 on news. We have just received official word that EUnet is now fully registered as a trademark in Denmark.

Other services: our tape distribution is going fine, about 40 tapes were shipped in the last half year. We have made a new contract with our administrator.

That completes the Danish story.

*Keld Simonsen*

# UKUUG News

*Mick Farmer*
*mick@cs.bbk.ac.uk*

*Department of Computing*
*Birkbeck College*

Mick is a lecturer at Birkbeck College (University of London) and the Secretary of UKUUG. His interest is in all aspects of Distance Learning and he is the Senior Consultant (Software) for LIVE-NET, an interactive video network connecting London's colleges. He is also a member of the University's VLSI Consortium, mainly because the design tools draw such pretty pictures.

## UNIX Security Workshop

We held a very successful meeting on 16th February at the Institute of Education, University of London, attended by over 150 delegates. Many aspects of UNIX security and administration were discussed, leaving many delegates with the wish for a similar workshop to be held in the near future. The proceedings were videoed and the edited programme is now available, initially, in the UK on a three-hour video cassette (details elsewhere in this issue).

## Pete, Farewell

Pete Collinson, the founder of the UKnet network, leaves 'ukc' (The Computing Laboratory, University of Kent) after 15 years on 31st May 1989. He started the network as an informal method of communication between early UNIX afficianados in the UK. This was part of a wider interest shown by a small group of similar-minded UNIX people in Europe, particularly in Holland. In the early years between 1980-1984 the number of communicating sites/machines in the UK varied between 20-50. Mail was paid for by each site and the structure was friendly rather than formal. Originally 'news' came in via a free link to 'ukc', supplied by an ex-student of Pete's. When this charitable route ceased in 1984 it became necessary to find an alternative supplier, which meant that 'ukc' was suddenly involved in IPSS charges for shipping Megabytes of news and mail on slow modems. A charging scheme was started and then the VAT inspectors became interested.

Fortunately for the UK Pete was not easily defeated. The technical problems in handling a UK domain naming system, that was unrelated to the rest of the world, the complexities of potentially changing nightly any of the world mail routings based on latest mapping information, and the authorisation/stopping and charging of mail and news were mere intellectual exercises for him. He will be greatly missed! The overall result is that, since 1985, UKnet has expanded linearly with no signs of abatement. The latest statistics are as follows:

| UKnet Statistics (April 1989) | |
|---|---|
| Academic Sites | 199 |
| NRS-registered Academic Sites | 182 |
| Non NRS-registered Academic Sites | 17 |
| Commercial Sites | 187 |
| NRS-registered Commercial Sites | 88 |
| Non NRS-registered Commercial Sites | 99 |
| Total Sites | 386 |

The continuing structure at 'ukc' will be:

Ian Dallas, PSS & JANET gateway problems, *ind@ukc.ac.uk*

Martin Guy, FTP problems, *mg@ukc.ac.uk*

Ian Harding, News manager, *ih@ukc.ac.uk*

Peter Houlder, Anything else, *uknet@ukc.ac.uk*

Pete is taking the plunge and becoming a freelance UNIX consultant, he can be contacted as *pc@hillside.uucp* or by phone on +44 227 61824. Pete's leaving has been accompanied by two other members of staff. Sean Leviseur (general mailer/networking knowledge) and Chris Downey (former News Manager). Sean is also doing freelance work and can be contacted as *sjl@westhawk.uucp* or by phone on +44 9282 2574. Chris will be working for Unisoft in London, phone +44 1 315 6600.

Ukc will continue to support the network and we have many interesting ideas for the future, but members are requested to show patience in the early days of transition.

## UKUUG Summer Meeting 1989

This meeting takes place in Glasgow, Scotland, on 27-28th June. The local organiser is Jim Reid from Strathclyde University. A synopsis of the papers organised so far is given at the back of this issue. Jim Crammond will also organise a tutorial on the UK-Sendmail package.

## UKUUG Winter Meeting 1989

We plan to hold this meeting in Cardiff, Wales, on 12-14th December, 1989. Further details will follow when they are available.

## UKUUG Summer Meeting 1990

We plan to hold this meeting in London, England, on 11-13th July, 1990. Further details will follow when they are available.

## UKUUG Winter Meeting 1990

We plan to hold this meeting in Cambridge, England. Further details will follow when they are available.

## London UNIX User Group (LUUG)

*The X-Windows system de-mystified* was the title of Ray Anderson's talk at the London UNIX User Group in April. More than 60 people crowded into the lecture room for the latest LUUG Lecture. Ray Anderson (IXI Ltd.) opened the talk by describing the origins of X in MIT's Project Athena; a far-reaching programme intended to help students and tutors communicate through computers. The goals of X were to be device and network independent, to support several concurrent applications on each display, and to be extensible.

In 1984 Robert Sheifler of MIT started with a copy of 'W' from Stanford and by July 1986 the X10 tape was made available to the world. Sun's announcement of their NeWS windows system later that year prompted eleven major companies to join MIT in forming the X Consortium. With a full-time team now at work, X11 was produced in less than a year. X11 is the current version, and has now reached Release 3.

X is based on the 'client-server model': client programs communicate with a server process which handles the screen, keyboard, and mouse. To people used to 'fileservers', 'printservers' and the like, it seems a little odd to have the *server* on the desk while the *clients* are in a back-room somewhere. "No problem" said Ray: "Servers exist to look after scarce resources. By comparison with the number of tasks they handle, *users* are getting pretty scarce!"

Some important features were described: The X-window system (unlike Presentation Manager) has an *asynchronous* protocol, which makes it usable over high-delay links such as satellite channels. In fact, X could be made to work over *any* reliable order-preserving channel. The X server maintains a hierachy of windows, and the protocol is designed to make window-creation cheap. As a result, almost every rectangle on the screen is a window in its own right. Even the humble scrollbar is likely to contain four or more windows.

The lecture ended appropriately with a set of slides showing the range of available widget styles and window managers.

Ray Anderson can be contacted at:

IXI Ltd. of Cambridge.
Phone: (+44) 223-462131

IXI distribute the X-windows system, the programming manuals, and also their own X.desktop product.

The next LUUG lecture will be at 19:00 on Thursday 25 May at 18 Bedford Square, London WC1. Jim Crammond of Imperial College will describe the new UK Sendmail Configuration Package.

The LUUG lectures are organised by: Andrew.Findlay@brunel.ac.uk

# UKUUG UNIX Security Workshop

*Thomas Grossi*
*grossi@capsogeti.fr*

*Cap Sesa Innovation*

## Introduction

I was able to attend the UKUUG seminar on the Security of UNIX Systems. There was a diversity of interesting presentations, whereas the audience was almost all British, and I think that I was the only delegate from France.

*The Hackman project*
*Russell Brand, Lawrence Livermore National Labs*
*speaking for Peter Shipley*

The Hackman project goal was originally to give users a safe system. Originally they planned to catalogue all security bugs, but that turned out to be too vast a domain, so they restricted it to UNIX bugs, and thereafter to BSD UNIX. The end result of the project will be a book listing known security bugs, describing them, giving a history and an program illustrating the problem, and showing how to fix them. Another goal is to come up with a page or two of programming hints to help ensure that you write a secure program (like ''never use gets; use fgets instead''—at the heart of the recent Internet worm).

Brand has an interesting definition of a security bug: something that lets someone do something you wouldn't want him to do if you knew he could. All bugs, says he, are based on erroneous beliefs at some level or another – the programmer assumed the system would work a certain way, but it doesn't. Some bugs from version 4.1a still exist in SunOS 4.0.2. Sun's Yellow Pages have turned out to be a veritable mine of bugs: for the past few weeks, they've been coming up with a new bug each day.

One of the questions from the audience concerned System V. Brand replied that the group had picked BSD because it was what they were all most familiar with and what they used the most; if there is a great deal of overlap between the two, it might turn out that for a small amount of additional work they could cover SysV as well, but they don't know.

Another question concerned the possibility of giving the catalogue to vendors in exchange for a binding agreement in which the vendors agreed to fix all the bugs within a certain period. Brand said that they had not thought of that yet, but seemed pleased by the idea.

Brand himself was a major source of questions for the rest of the day.

*666 etc.,*
*Lindsay Marshall*
*University of Newcastle-upon-Tyne*

The title of this talk was a play on the default privileges of files on UNIX systems and on caballistic symbolism (remember Proctor & Gamble?). You should come away from this talk with the feeling that protection=666 is something to be wary of!

An awful lot of security holes exist for the simple reason that many system files have too-permissive permissions. A major problem is that vendors dump the machines with absolutely no explanations on users who just don't know what they're doing. For example, the architecture department at Newcastle recently brought a workstation onto the network that had no root password and read/write permission on everything!

Marshall suggests that a multiplicity of distinct uids and gids for different utilities (uucp, news, sendmail, yp, etc.) will tighten up considerably a system's security. The SAs should su to each specific utility-user rather than working as root. This has two other significant advantages: 1) the amount of damage you can do accidentally as one of these users is thus greatly diminished; 2) it allows you to farm out duties to several people. There is somewhat of a problem in figuring out how to can carve the utilities into separate uids, but it's less difficult that one would expect.

Other random comments: There's no reason whatsoever for allowing a binary file to be readable—to run a program it suffices to be executable. One should keep an eye on setuid programs—they're inherently dangerous. 'ps' and 'who' could be considered security problems—Joe User shouldn't necessarily be able to tell what everyone else is doing on his machine. When using chmod, type `chmod a+rw` rather using the format `chmod 666`—it's less error-prone (since it's more mnemonic and won't remove execute permission by accident). Watch out for gnu software—a lot of it has a umask of 000 (and therefore leaves files with a protection of 777). This is understandable if you take into account Robert Morris' character.

*Secure RPC*
*Andy Rutter*
*The Instruction Set*

W.r.t security, network applications are hard. Security is often ignored or added as an afterthought, and credentials can be easy to fake. As a result, it's usually left up to the applications to take care of security issues. Of course, the same comments often apply here as well, with the additional problem that the solutions are usually ad hoc.

In SunOS 4.0, it was decided to offer authenticated platform services, to assure at least a minimal level of security. The goal was to make the network as secure as a unix time-sharing environment (adequate for general use) while adding a minimum of overhead. It was assumed that it is not possible to replace packets in transit through the network and that eavesdropping is not a problem; therefore the data are not encrypted. On the other hand, impersonation ('masquerade') is taken to be a problem, so servers and clients both want authentication of their dialogue partners.

How is this authentication done then? The client selects a conversation key (CK) at random. (This ensures that no one can break the encryption schemes by monitoring conversations and picking out the keys used.) This key is encrypted using a public key (PK; some well-known constant raised to a secret-key power). The client then sends a triplet to the server: (identifier, PK-encrypted CK, CK-encrypted time window). The server acknowledges by sending back the time value + 1, encrypted with the CK (again, 1 is added so that the value sent back is different from the value received—thus preventing eavesdroppers from figuring out the keys). If the validation sent back falls within the allowed time window, the client knows that it is talking to the authentic server.

How does this affect users of the Sun network? NFSSRC 4.0 offers secure NFS and YP. For an RPC user, the changes are completely transparent. Performance degradation is about 2 per cent. For the installer, there are some minor differences. Public keys are kept in /etc/publickey. In /etc/exports one must add:

```
/home-secure,access=net-group
```

and one must also add `secure` to the options in /etc/fstab.

Of course, it *is* possible to break secure RPC. One method is to record transactions that occur on the network and replay them when the server crashes. This can be avoided by making the time window less than reboot time. (By default the window is a half-hour or an hour.) If there is a gateway, the network automatically becomes insecure, because the gateway has the possibility of substituting data packets. Rlogins can be a Trojan horse. Also, if someone is dedicated enough, it is not difficult to jam the network and substitute data packets.

Other problems arise as well. When a server crashes, the cached keys disappear and conversations cannot be resumed—they must be restarted. If a server is diskless, its disk server can be faked. Setuid programs do not work unless the user is logged in, as the conversation key is generated at the beginning of the session. And you must be able to trust your public key servers.

*YP*
*Willian Roberts, Queen Mary College*

The Yellow Pages was originally intended to be a distributed look-up service, not a distributed data base. This is why it fails so badly when it comes to modifying the data it stores.

How does YP work? The system has been set up so that it's completely transparent to application programs. Whenever an application needs some information that is not known locally, the routines in libc know to ask the ypbind.

First, a bit of background information. On each machine a program called 'portmap' keeps track of rpc services that have been registered, and tells which ports are used by which servers. In general the ports are given out arbitrarily, but for YP servers, the server is given port 111. When ypbind needs yellow page information, then, it broadcasts a pre-request to port 111 on all machines. This message says, "say 'yes' if you can answer this, else say nothing." (In this manner ypbind can act on the first answer it gets back.) Ypserve replies giving the port number on the local machine of a data base server corresponding to the request. Ypbind then sends the real request to the address indicated.

There are several problems with this system. First off, ypbind doesn't keep track of the number of attempts it makes to find a DB server. It basically keeps on trying forever, and if it can't find one, your process hangs. Likewise, libc only checks to see whether rpc has failed or whether it has replied "yes"; the "no" case has not been considered. The use of portmap is a major security hole: you can ask to unregister the official ypserve and put your own (bogus) server in its place.

*The Cambridge Experience,*
*Piete Brooks.*

This talk was not terribly enlightening. The blurb in the day's program sums it up quite well:

"Cambridge University's Computer Laboratory has a number of vendors' systems (DEC, SUN, HP, Xerox, Acorn & HLH) mostly running various flavours of UNIX. The central power is a 'processor bank' of Micro-Vax-IIs which live 'in the cellar'. They are used a s CPU servers from ethernet terminals. The main (disk) servers do not have user accounts (only admin) and are trusted by all clients. Access to and from WAN to LAN is transparent. We also have IP over X.25 links."

"When some anomalies were noticed, our academic laissez-faire approach was laid aside and we made some use of our security gurus to tighten things up. Passwd was changed to force eight mixed chars, no words or keyboard strings, etc. Login was tightened up to check the calling address and behave accordingly. The importance of reliable logging was once more emphasised to us."

*Some Myths and Facts about UNIX Security,*
*Mario Wolczko,*
*Department of Computer Science,*
*University of Manchester.*

A breach of security is possible when 1) a file has inappropriate owner or group permissions, or 2) a process attains an illegal user id. The former is usually due to improper user education; the usual method in the latter case is via setuid.

The kernel itself does not really care about users and passwords; for it, users and groups are only numbers. The problem lies in the applications built upon the kernel. For example, demons that trust everything they get, users who exchange passwords.

All processes have a real uid and an effective uid. The real uid is the uid of the login user. The effective uid corresponds to the permissions allowed to the process and is governed by the setuid bit. At any time, however, a process can set the effective uid to the real uid OR VICE VERSA.

The danger of setuid scripts: given a script 'chtty' that is run setuid, give four ways of grabbing root privileges. Its contents are:

```
#! /bin/sh
/etc/chown `whoami` `tty`
```

Solutions:

1.
   ```
   cp trojan whoami
   PATH=.:$PATH chtty
   #   trojan   gets   executed
   instead
   # of system whoami
   ```

2.
   ```
   cp trojan etc
   IFS=/ PATH=.:$PATH chtty
   # IFS is the field separator.
   # in this case, 'etc'
   # becomes an executable pg
   ```

3.
   ```
   ln /usr/bin/chtty -i
   -i # -i option gets passed to
   sh,
      # makes it interactive
   ```
4. there's a small time window between the instant at which the kernel has decided to give the process root permission and the execution of the file, during which it is possible to substitute another file for chtty and have it executed instead.

Another weak spot lies in the fact that file descriptors are inherited across execs, and can even be passed down sockets. (In SunOS 4, 'send' only works with the 'unix' call.)

*Workstation Security,*
*Jim Reid, University of Strathclyde*

A list of ways security can be broken on workstations.

Simple attacks:

- — the frame buffer is usually writable
- — window manager logs are usually readable (e.g., Sun's commandtool leaves mouse-selected data in /tmp)

Boot attacks:

- — by default, anyone can boot his workstation in single user mode
- — you can get a root login by interrupting the execution of /etc/rc
- — if you have a local streamer, you can load mini-unix from the installation tapes

Monitor attacks:

- — patch kernel data – proc table or user area entries
- — patch kernel text – patch the code with adb

  solutions:

  - disable monitor mode
  - get a secure PROM monitor

YP attacks:

- — corrupt the data bases

NFS:

- — the protocol is open and public
- — it's extremely vulnerable, due to its weak authentication
- — even SunOS's 'secure' NFS can be broken the attacks:
- — save up NFS actions, since they contain file handles, and replay them
- — one can forge file handles, NFS credentials, mount request and even bogus clients

  protection:

  - use nobody's uid (-2) for root requests across the network
  - setuid, sgid mount options
  - secure UDP port numbers (in NFS = 2049) – any user can bind to it if it's not running
  - mound demon checks: set ipaddr_check to non-0

Advantages of SunOS 4:

- — controlled root access
- — 'secure' option similar to secure RPC (above) with stronger authentication and public-key encryption. However, the keys are distributed via YP!!!

*The Internet Worm*
*Jim Reid*

I won't go into the details, as they are available in the various reports that are circulating .

In his comments, he stressed how vulnerable we would be to a virus that would be posted to comp.sources.unix, for example. In a scenario of what might happen, he depicted a virus for libc that would be included in some utility and which would lie in wait until the program ran as root. When the unwitting root came along, the utility would then install its virus in the system's libc, for example. Consider the difficulty of finding such a virus! Or perhaps it could install a faulty login. But there you might find it out by looking at the code, so better yet, install a faulty C compiler that checks to see if it's compiling the program 'login'. Then when one looked at the code, it would be ok, but the virus would still be there. But one might be able to find its source by looking at cc with a debugger. So you could modify the debugger, so that it knows what not to display if it gets called on cc ...

*Open Session,*
*at the day's end.*

The question was once again raised, How many system's actually pay attention to frequent password changes and to the security of those passwords? Another interesting point made was that dump tapes should be secure, else anyone can procure a copy of sensitive files.

Most of the talk, however, centered around students in a computing environment. It appears that most security problems come from students who find that break-ins are a clever thing to do. The most effective approach to dealing with this situation seems to emphasise the ethics involved, promising appropriate punishment but not making a big deal out of the problem.

*This is an article that was originally printed in the AFUU Newsletter Tribunix, reprinted by kind permission.*

# Networking Column

*Peter Houlder*
*uknet@ukc.ac.uk*

*Computing Laboratory*
*University of Kent at Canterbury*

Peter Houlder has been in the Computing Laboratory at the University of Kent for the last 4 years and looked after day to day UKnet admin work in the last 3 years.

He graduated in Geography from Kings College, London in 1970 and then spent 9 years in business—dropping out in 1979. He then spent a year touring North, Central, South, and Carribean America, became interested in archaeology and spent three years exclavating in Britain and Europe.

Two Masters degrees, the first in Archaeological Sciences and the second in Computer Science, followed in successive years. Maggie in the meantime reduced archaeological funding, so he arrived in 1984 kicking and screaming into the world of Computing. He has since got to quite enjoy it.

He is married with two labradors.

This month we have two reports. The first from the Italian and the second from the Norwegian Network. Spain have promised an article next time, which should be interesting as they have gone from 0 to 31 sites in about a year. Denmark have pointed out that they have submitted lots of details in the past. My target list is therefore reduced to Greece, Ireland, Iceland, Luxembourg, Portugal and Yugoslavia.

## Status Report from the Italian Backbone

*Joy Marino*
*Alessandro Berni*

*Dipartimento di Informatica*
*Sistemistica e Telematica*
*Universita' di Genova*
*iunet@dist.unige.it*

### The IUnet

The network continues to grow: our sites count is currently 46 and we plan to increase by another 5-6 in the near future.

10 sites are receiving the news: this causes us problems with the 2 dial-up lines (1200-2400 bps), which are always busy, and with disk space (only 170 Megabytes). Another 2 telephone lines are going to be ordered to support the Dataconsyst T2000 modems (that is Telebit Trailblazers in a different box :-) ) that we have just purchased. To conserve disk space we run expire each night and move the old news on a Sun 4 server as a temporary solution.

Our 'news' plan is the following: as soon as it is possible move the WORM disk system from the small machine based on a National processor to the faster Sun 4 (a.k.a. ugdist) in order to 'see' it using NFS. Then, install the UNO server (USENET News on Optical disk) developed here at DIST and distribute the remote query program to our nodes.

This will prevent us from having to make periodic backups on tape (since all the news of interest will be maintained online on the 2 Gigabytes optical disk) and will enable our users to 'order the back issues' making use of the keyword search service implemented by UNO.

We plan also to have an on-line archive of PD software, based on the same WORM disk with NFS.

What we really hope anyway is to get a faster machine with SysV.3, 200-300 Megabytes of disk space and NFS support. It seems that UNISYS will be ready to give us a machine matching our needs before the *i2u* Convention (taking place in early June).

A Cisco gateway (MGS with a 2200 packets per second board) has been purchased by the DIST to support a *private* IP link to the CNUCE Institute of the National Research Council, which is connected to the USA using the old SATNET. It can however support a second synchronous link, so it has been decided that it will serve also the incoming EUnet IP link. This link to Pisa will be used to reach the other italian networks that is to exchange .IT mail.

What we would like to be made clear is that even if we were to get the link to the Internet via the CNUCE today, we would have still the same determination to pursue the establishment of an European IP network and are ready to give full support to this initiative.

We have contacted the INFNET (Italian HEPnet) people about sharing a small part of their Bologna-Geneva high-speed link but got no answer until now (maybe they are too scared :-) ). We are however determined to go on with this matter and in case the 'INFN Connection' does not come to a solution we will be forced to hook up to some European backbone. That would be a pity, since we 'backboners from the suburbs of the network' would be forced to pay a higher price for the same service (our distance from 'central' Europe raises the line cost, which is already high since our PTT has extremely expensive fares).

An 'IP Network(s) Interconnection Demo' is planned for the *i2u* Convention.

## IUnet and EUnet

We are currently maintaining links with the following backbones: mcvax, inria, unido, enea, dkuug. The problems with our X.25 software have been finally solved, so we would like to establish links with the other backbones (possibly in both directions, that is we would like to *call* and to *be* called).

**Mail statistics within IUnet (Feb 88-Jan 89)**
(in Kbytes)

| site | (ita) | (bit) | (eur) | (usa) |
|---|---|---|---|---|
| automa | 3.96 | 0.00 | 0.00 | 0.00 |
| ccaumi | 1993.41 | 531.91 | 877.29 | 725.25 |
| cip | 17.56 | 37.51 | 123.31 | 48.21 |
| cise | 242.08 | 10.64 | 1645.47 | 352.39 |
| cnriac | 48.95 | 6.85 | 5.73 | 21.36 |
| crai | 33.12 | 18.45 | 619.59 | 309.72 |
| crcc | 31.70 | 0.00 | 7.85 | 7.34 |
| cselt | 5240.00 | 8.57 | 2035.00 | 1472.19 |
| delphi | 775.18 | 43.05 | 2927.00 | 14813.54 |
| depolimi | 0.00 | 0.00 | 0.88 | 4.17 |
| dipisa | 305.73 | 375.38 | 2295.00 | 2173.06 |
| disrm | 138.87 | 25.02 | 1472.31 | 1482.18 |
| enidbo | 174.93 | 197.04 | 1159.64 | 569.10 |
| i2unix | 3881.00 | 950.30 | 4213.00 | 4085.76 |
| ic8u74 | 15.95 | 7.00 | 109.94 | 319.98 |
| iconet | 6345.13 | 53.79 | 5938.00 | 594.78 |
| ieeupdb | 7.26 | 0.00 | 11.97 | 29.11 |
| imacnr | 0.71 | 0.51 | 5.76 | 240.47 |
| intecs | 0.00 | 0.00 | 1219.00 | 0.00 |
| irst | 1095.00 | 1150.00 | 3756.03 | 2239.41 |
| leonardo | 708.49 | 0.77 | 622.00 | 186.21 |
| lipr | 5.54 | 0.00 | 0.00 | 0.00 |
| littix | 4.42 | 0.00 | 0.81 | 0.29 |
| olhqm01 | 2.96 | 1.26 | 47.42 | 6.39 |
| olita2 | 0.00 | 1.26 | 0.00 | 6.39 |
| sadix | 6.72 | 3.44 | 3.73 | 26.07 |
| sarinit | 106.75 | 0.41 | 1323.67 | 8759.53 |
| semto | 6611.49 | 35.59 | 946.67 | 7969.23 |
| siemil | 35.33 | 0.00 | 0.00 | 0.00 |
| sissa | 0.00 | 9.68 | 0.00 | 6.38 |
| sogesta | 54.00 | 0.00 | 0.00 | 0.00 |
| synarea | 3.07 | 0.00 | 1.39 | 1.93 |
| ubdm | 752.68 | 294.15 | 587.17 | 810.07 |
| udsab | 338.62 | 1964.93 | 818.00 | 2711.86 |
| ugdibe | 0.88 | 0.00 | 88.50 | 0.00 |
| ugdist | 1371.15 | 220.78 | 2507.00 | 2663.34 |
| unirel | 1.46 | 0.00 | 0.00 | 0.00 |
| unisysit | 0.00 | 0.00 | 0.00 | 6.39 |
| zel | 86.36 | 0.00 | 30.08 | 23.31 |
| Total | 30440.46 | 5948.30 | 35399.19 | 52665.38 |

## IUnet and the Other Networks

Talks have been going on within the GARR (Group for the harmonisation of research networks): the immediate result is that the IUnet subdomains have been registered in the primary server for .IT (making use of MX records). We now *pass* on our UUCP link to the CNUCE a rapidly growing volume of traffic. When we decide that it's the right moment (that is, when they will depend on our mail service more than they do now) we'll inform them that we cannot afford any longer to pay *with our money* for *their* mail. Our hope is to find a satisfactory agreement on this *financial* topic, since we are sure that they can afford it :-). The problem with mail directed to .IT that is being bounced back to the sender is due mostly to problems with the routing tables of the other italian networks: cooperation is going on to ensure that this problem finds a rapid solution.

### IUnet and the Hackers

Most of you know of the campaign set up by a group of wily hackers (I mean *cyberpunk hackers*) who claimed they could offer the equivalent of our mail and news service for free.

Since they refused to subscribe the EUnet services they had been *blackholed*, that is their mail was being rejected and returned to the sender. Unfortunately this was not well understood on the other side of the Atlantic: site pyramid (willing to support an initiative that seemed to be compliant with the 'Usenet spirit') offered them a news and mail feed for free. The danger of having an anti-EUnet growing day-by-day was avoided when pyramid disabled their UUCP logins: an investigation with Italcable had proofed that those hackers were using stolen PADs and X.28 passwords (that's why they offered their service for free!) a real hacker's solution!

After that they have tried (are trying) to re-establish friendly relations with us: we of course feel the need of granting to *amateurs* a *limited* access to the network, but we don't think that those hackers of Sublink have the necessary accomplishments of capability and honesty to run the *amateur* part on our behalf.

A *popular computing* magazine, which already runs the biggest Bulleting Board System in Italy (based on a UNIX system b.t.w) has shown a definite interest for *retailing* EUnet mail and news to their end-users.

## EUnet in Norway

*Steinar Overbeck Cook*
*steinar@fdmetd.uucp*

*Fellesdata A/S,*
*Nedre Skoyenv 26,*
*Postboks 248,*
*N-0212 OSLO 2,*
*Norway*

Here is some information on how things are going in Norway.

Norway has officially been part of EUnet for approximately 1 year now. We have had various problems with our backbone, a Norsk Data machine running some sort of UNIX under SINTRAN. This did not work very well so Norsk Data has 'given' us a new backbone, a Wyse 386 PC, which is running UNIX from Interactive. We are still having some troubles though, caused by modems not responding, telephone switchs which does not switch and a lousy serial port controller with brain-damaged driver software. These problems only apply to the members dialing into the backbone, all communications with the mcvax and so on, seems to work very well. The NUUG's network group are working hard to solve these problems. They are now going to try another controller to see if that helps. The modems on the backbone has also been changed.

When our problems with connecting to the backbone are solved, we will start testing Trailblazer modems. It is very expensive to use dial-up lines in Norway so we hope that faster modems will attract more members to use the services of EUnet.

# Puzzle Corner

*Mick Farmer*
*mick@cs.bbk.ac.uk*

Hi peeps,

A number of you have pointed out that Puzzle Number 1 (and other puzzles of that ilk) can be solved using the programming language Prolog. Here is one version:

```
/* Dedications - EUUG Puzzle Number 1 */

select(H,[H|T],T).
select(X,[H|T1],[H|T2]) :-
    select(X,T1,T2).
authors([book(kernighan,steve,C1),
         book(lesk,brian,steve),
         book(ritchie,ken,mike),
         book(bourne,dennis,C2),
         book(thompson,D1,C3)
        ]) :-
    select(D1,[mike],_),
    select(C1,[brian,ken,dennis],T),
    select(C2,T,[C3]).
fact1(L) :-
    select(book(thompson,_,X),L,_),
    select(book(_,X,brian),L,_).
fact2([]).
fact2([book(_,D,C)|T]) :-
    +(=(D,C)),
    fact2(T).
solution(X) :-
    authors(L),
    fact1(L),
    fact2(L),
    select(X,L,_).
```

There isn't space to describe this program in detail, but if anyone has a different version I'd like to see it. Any comments are also welcome.

Now on with the solutions to the puzzles in the last issue. Puzzle Number 2 is a thinly disguised problem in number bases. The number 1,000,000 in base 10 yields 11333311 in base 7. Thus, working from the right, there was one rebate of 1 ECU, one of 7 ECUs, three of 49 ECUs, three of 343 ECUs, three of 2,401 ECUs, three of 16,807 ECUs, one of 117,649 ECUs, and finally one of 823,543 ECUs. I wonder which national group got that!

Puzzle Number 3 can be solved without the need for complicated arithmetic or the value of $\pi$. The ball of wire is 24 inches in diameter. A cylinder 24 inches in diameter and 24 inches long contains half as much again as our ball. Therefore a cylinder 24 inches in diameter and only 16 inches long has the same contents as our ball. We can consider this cylinder as a large number of thin wires (one-hundreth of an inch in diameter) and all 16 inches long. There are $24 \times 24 \times 100 \times 100 = 5{,}760{,}000$ of these giving a total length of 92,160,000 inches. Thus the wire is 1,454 miles 2,880 feet in length.

Puzzle Number 4 first appeared in the Sunday Times in 1968. My prolog system expires (out of stack space) when faced with this one:

The seven dwarfs always walk to work in single file and in the same order: Happy, Doc, Sneezy, Bashful, Sleepy, Grumpy, and Dopey. At holiday time two stayed at home, the others from habit went in single file, but never in their working order of march.

One day, five of them set out to build a new house for Snow White, each carrying something useful. She had given each dwarf a different coloured scarf, and Dopey was very proud of his green one, but Bashful would not show anyone his. Sneezy marched immediately ahead of the one carrying the twine. The saw bearer was followed by white scarf.

At the site it was found that no one had brought nails and no one volunteered to go back for them. Blue scarf told Grumpy to go, but Grumpy refused and told him to send the one with the hammer, adding that, as leader for the day, he should have thought of that himself.

The other three gave their opinions. Happy said the one with the twine should go. White scarf said it should be Doc. The one behind him, who today was ahead of two who were normally in front of him, said red scarf should go. Sneezy then said it was daft to suggest sending Dopey and a quarrel began.

When calm was restored, it was found that No. 3 and No.5 on the day's march had suggested the same person, so he was made to go.

Who was he, and what was he carrying?

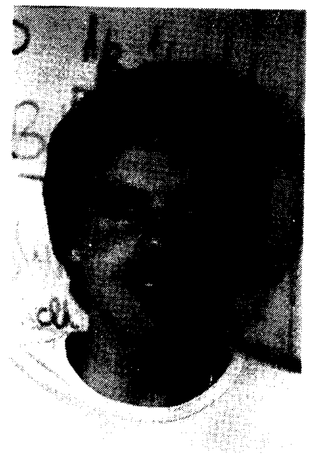
# UNIX CLINIC (*aka* Call Doc Strange)

*Colston Sanger*
*doc.strange@olibc1.oliv.co.uk*

*Olivetti International Education Centre*

*with a special guest appearance by*
*Sam Nelson*
*sam@uk.ac.stir.cs*

*Dept of Computing Science,*
*University of Stirling*



Colston Sanger is a lecturer at the Olivetti International Education Centre, Haslemere, UK and a visiting lecturer in the Faculty of Engineering, Science and Mathematics at Middlesex Polytechnic. Nothing much else to report this issue—apart from a trip to Japan. *Ah so deska!*

Sam Nelson escaped briefly into the unreal world of commercial software production (no names, no pack drill) but realised later that it was only a sabbatical appointment. Returning to the fold, he conned some trusting souls out of their superusers' passwords and now divides his time disproportionately between mediocre bass guitar playing, erratic photography and the Intuitionist school of UNIX (system) administration (as in 'Well, it seemed like a good idea at the time...').

## The Use and Abuse of Regular Expressions

**Dear Doc Strange,**

**Whenever I start and people look at me in a funny sort of way. Is it my expressions?**

**Perplexed of Putney.**

*Dear Perplexed of Putney,*

*Hmm... Yours is indeed a difficult case, demanding much thought and accumulated wisdom ... methinks I will have to call upon a real expert...*

*Ladies and gentlemen, a big hand please for your special guest contributor—all the way from Stirling in Scotland—Mr Sam Nelson! Taran-ta-ra!!!)*

The standard UNIX[1] System V manual set contains upwards of a dozen command entries which use string regular expressions in one way or another. It never ceases to amaze the author how many UNIX[2] users on his site know very little about their use. Based on the notions that this is probably the case at many other sites across Europe, and that there are at least some real UNIX users reading this journal nowadays, it seemed worthwhile to describe some of the marvellous things that can be achieved with regular expressions.

Two or three minutes spent working on the right regular expressions can sometimes save several hours editing, condensing an enormously tedious session into perhaps half a dozen commands or fewer. The generalisation of the regular expression system under UNIX System V into C library function calls opens up new opportunities for the simple construction of regular-expression-based applications.

## Regular Expression Syntax

For a complete explanation of the syntax of the ordinary UNIX regular expression, there is no better source than the first three pages of the `ed(1)` standard manual entry. Some tools extend or modify this syntax for their own purposes, but the general principles below should be enough to understand what follows:

- A single character in a regular expression string matches itself in the target string. 'Special' characters are generally matched in the usual UNIX way, by prefixing them with the backslash character '\', although there are occasional pathological situations.
- A list of characters in square brackets matches any one of the characters in the list. A shorthand form is also available, thus the list `[0123456789]` can be written as `[0-9]`.
- A dot '.' matches any single character. A string of single character matches followed by `*` matches any arbitrary list of the single characters it would have matched on its own, including a zero-length string of them.
- A `^` at the beginning of a regular expression matches the beginning of a line; a `$` matches the end of a line.

Thus `[+-]*[0-9]*` matches any decimal integer; `[fF][a-zA-Z]*` matches any word whose initial letter is 'f'; and `[a-zA-Z0-9_!%]*@[a-zA-Z0-9_.]*` matches *almost* any network mail address currently in use.

The above description gives the impression that an attempted match can give only a straightforward 'yes' or 'no' answer. While this sort of result is important, it only scratches the surface. The UNIX editors, `ed`, `ex/vi`, `sed` *etc*, as well as `grep` and `awk` use this to identify lines in files (no coverage of `awk` is attempted here: there is already a book on the subject!).[3] The real power becomes apparent when the regular expression special brackets \( and \) are used in a regular expression string to identify and use interesting parts of the matched string. In the simplest case, `expr` returns a single bracketed substring as its standard output if the match succeeds, so

```
    expr arbitrary_integer : '[+-]*[0-9]*\([0-9]\)'
```

provides an alternative, quaint method to generate remainder on division by 10 (not recommended for general use!) and

```
    expr pathname : '.*/\(.*\)\..*'
```

is a generalised implementation of `basename` which strips *any* suffix from a pathname rather than just a

1. UNIX is a registered trademark of AT&T in the USA and other countries.
2. The author is firmly convinced that 'UNIX' is a noun.
3. Aho, Kernighan and Weinberger, *The AWK Programming Language*, Reading, Mass., 1988 (Addison-Wesley, ISBN: 0-201-07981-X).

single specified suffix. For example, given a pathname of `/staff/colston/lib/junk.old` it will return just `junk`. `ed` allows the identification of a number of substrings in its 'substitute' command for use in the 'replace' string, so

```
s/\([^ ]*\) \([^ ]*\) \([^ ]*\)/\2 \1 \3/
```

transforms 'UNIX operating system' into 'operating UNIX system' (note here that a ^ as the first character inside square brackets causes a match with any character *not* present in the list, rather than a match with those that are present).

## A real-life example

Frequently the 'glue' that ties together the regular expression operators is as important as the regular expression operations themselves. Consider the following scenario (this was a real-life situation, unfortunately!):

Due to an edict from somewhere above, the order of two parameters in a particular library subroutine has to be changed. This being a very popular and general routine, it is called in dozens of places in the huge tree of source files for which you are responsible. By hand, this requires you to search every source file with the text editor one by one, and actually type the new function call every single time because the parameters used vary widely. One possibility for a mechanical method is:

1. Create a list of files which contain calls of the relevant function (one could simply perform the operations below on all files in the tree, but this efficiency measure is reasonably cheap):

   ```
   find . -type f -exec grep 'fn(.*)' {} \; -print | grep '^\.'
   ```

   (If your `grep` can be made silent, the second `grep` is unnecessary.)[4]

2. For any file in a given list, swap the order of the two parameters in any call of the given function:

   ```
   sed 's/fn(\(.*\),\(.*\))/fn(\2,\1)/g' file > temp_file
   mv temp_file file
   ```

3. Combine the two...

   ```
   for f in `find . -type f -exec grep 'fn(.*)' {} \; -print | grep '^\.'`
   do
           sed 's/fn(\(.*\),\(.*\))/fn(\2,\1)/g' $f > t
           mv t $f
           echo "fixed $f"
   done
   ```

Unfortunately, in real-life this operation had to be carried out on a VMS system with no pattern-matching editor capabilities, and with extremely limited facilities for the creation of a list of files in a tree with a given property. It took almost a day.

## Yet another example

This one is not human-language-independent. It would probably fail miserably in German, at least. Suppose you have the text of a book on disk (say 3-400 pages) and your publisher wants a checklist of possible trademarks for crediting. The non-mechanical method is to read the entire book while holding a highlighting pen. A regular-expression-based approach might go as follows:

Assume the text of the book is contained in all of the TEX source files in the current directory:

4. In SunOS, this is the `-s` option.

1. Create a list of words in the book, separated by newlines:

   ```
   cat *.tex | tr -cs '[A-Za-z]' '[\012*]' | ...
   ```

   (A better definition of a word could easily be substituted.)

2. Remove from the list all words that aren't proper names and strip out duplicates:

   ```
   ... | sed -n 's/[A-Z][a-zA-Z]*/\1/p' | sort | uniq > propernames
   ```

   The list that remains will probably contain more than just the trademarks that need to be credited, but at least the body of text to be searched will have been reduced by a couple of orders of magnitude. The point is that so long as the approximation to the required list is larger, *but not too much larger* than the required list itself, hours of tedious proofreading disappear in a puff of shell script.

## Regular expression applications

There are certain weaknesses in the array of regular expression weaponry provided. Consider the following:

1. Suppose you want a list of 'GECOS fields' in the /etc/passwd file that contain a particular pattern. This can be done either by

   ```
   grep '^[^:]*:[^:]*:[^:]*:[^:]*:pattern' /etc/passwd | cut -d: -f5
   ```

   or perhaps by

   ```
   cut -d: -f5 /etc/passwd | grep pattern
   ```

   The fact that a pipe must be used here is a pity, because the task is a fairly straightforward one and on a large site the job is not going to complete in microseconds.

2. Take a fairly huge, flat directory of PXL, GF and PK format font files as used by $\TeX$ and its support tools, and reorganise them into a tree separated by both file type and magnification. Assume the filenames start out in the form

   *<fontname>.<magnification-number><type> e.g.*, cmr10.1500pxl

   and are to be deposited in a tree in the form

   dvifonts/*<type>*/*<magnification-number>*/*<font>.<type>*

   This may sound a little artificial, but it makes a lot of sense, since file access times can be decreased significantly by using sets of small directories rather than a single large directory. `expr` is underpowered for this task, since it can only return one substring at a time. The preferred method with the provided facilities is probably:

   ```
   for f in *
   do
   NF=`echo $f | sed 's@\(.*\)\.\([0-9]*\)\(.*\)@dvifonts/\3/\2/\1.\3@'`
   mv $f $NF
   done
   ```

   Here, a large number of more or less trivial pipes has had to be used where they really ought to be unnecessary, slowing down the task quite considerably. (`sed` allows any character to be used to bracket match/replace pairs so I have used `@` instead of `/` in an attempt to make it less confusing.)

The `PW` library provided with System V implementations makes available `regcmp(3X)` and `regex(3X)` for the compilation and execution, respectively, of regular expressions inside C programs. Unfortunately, no library function is provided for the substitution into a result string of substrings identified by `regex()`. The construction of such a function, which more or less implements the mechanism familiar to users of, say, the `vi` 'substitute' command, is relatively straightforward. The code for `regsub()` follows:

```
#define EOS      '\0'
#define PARTSUB  '\\'
#define WHOLESUB '&'

char *regsub(src,orig,rret,res,rsize) char *src,*orig,**rret,*res; int rsize;
{
  /*
   * Substitute substrings into a template.  Arguments are as follows:
   *   src:   the template string.  If '\digit' appears, it means 'substitute
   *          here the substring given as element 'digit' of the substring
   *          array 'rret' (qv).  If '&' appears, it means 'substitute here the
   *          whole of the original string'.  If '\char' appears, it means
   *          'substitute here the given character' to allow for '\' and '&'
   *          in the result string.
   *   orig:  the 'original' string.  If 'regsub()' is being used as part of a
   *          regular expression match/replace mechanism, this is the test
   *          string given to regex() to attempt a match against a regular
   *          expression.
   *   rret:  an array of substring pointers.  If the standard System V 'PW'
   *          version of regex() was used, this will have ten elements max, and
   *          the substring substitution mechanism can currently only cope with
   *          ten, as it uses a single digit as the index.  Obviously, all ten
   *          elements don't have to be filled in---you may not have used any!
   *          Reference to an unfilled substring is, in the best traditions of
   *          such things, 'unpredictable'.
   *   res:   optional buffer for the result string.  If NULL is passed,
   *          'regsub()' attempts to malloc() space for the result.
   *   rsize: The size of the buffer passed in 'res'.
   * The return value of the function is a pointer to the result string, if
   * space was found to accommodate it, or NULL otherwise.  Note that even in
   * the case of a buffer being passed in 'res', the address of this buffer is
   * still returned as the function result.
   */
  char *s,c,*ret,*rt,*osrc=src; /* Address saved for the second pass */
  char *malloc();
  int count=1;  /* Initialised to 1 to count the EOS character */

  /*
   * First, zip through the 'source' string, and calculate how long the result
   * string is going to be. This is easy because all the components are
   * available.  It also saves problems with running out of space in the middle
   * of a substitution, and makes error returns much cleaner.
   */
  while(*src) {
    switch(*src++) {
    case PARTSUB:        /* '\' char---followed by a digit, or not? */
      c= *src++;
      if(isdigit(c))
        count+=strlen(rret[(int)(c-'0')]); /* Add length of substring */
      else
        count++;         /* '\char' just means 'char' */
     break;
    case WHOLESUB:       /* '&' char means 'put whole orig. string here' */
      count+=strlen(orig); /* Add length of original string */
      break;
    default:             /* ...otherwise it must be a 'normal' character */
      count++;
    }
  }
/*
 * Now, sort out where to put the result string when it's built.  The
 * space required is in 'count'.
 */
if(res!=NULL) {        /* Test to see if buffer was offered */
  *res=EOS;            /* Tidy up in case of failure */
  if(rsize<count)      /* Is the buffer big enough? */
    return(NULL);
  else
    ret=rt=res;
}
else if((ret=rt=malloc(count))==NULL) /* Try to get a buffer big enough */
  return(NULL);
else
  *ret=EOS;
src=osrc;              /* Go back to the start of the source string */
/*
 * Next, perform the substitution.  This is the same as the size calculation
 * above, except that the characters are stuffed into the result string
 * instead of just being added to a counter.  There are probably some clever
 * optimisations available here due to having just skipped through the
 * string once, but the code looks cleaner this way, and it isn't exactly
 * a CPU-killer anyway.
 */
while(*src) {
  switch(c= *src++) {  /* Copy taken of each character, this time */
  case PARTSUB:
    c= *src++;
    if(isdigit(c)) {
      s=rret[(int)(c-'0')];
      while(*rt= *s++) /* Copy the substring into place in the result */
        rt++;
    }
    else
      *rt++=c;         /* '\char' means 'char' */
    break;
  case WHOLESUB:
    s=orig;
    while(*rt= *s++)   /* Copy entire original string into place */
      rt++;
      break;
  default:
    *rt++=c;           /* ...otherwise it's a 'normal' character */
  }
}
*rt=EOS;               /* This may or may not be missing... */
return(ret);
}
```

Now, given `regcmp()`, `regex()` and `regsub()`, the program `reg` which takes source string, regular expression and result template turns out to be about 30 lines of C:

```
#include <stdio.h>
#include <string.h>

#define RETNUM 10
#define RE_SOS "^"
#define RE_EOS "$"
#define RETURN_STRINGS ret[0],ret[1],ret[2],ret[3],ret[4],\
                       ret[5],ret[6],ret[7],ret[8],ret[9]

char *progname,*ret[RETNUM];

main(argc,argv)
int     argc;
char    **argv;
{
  char *test,*regexpr,*template,*comp,*result=NULL;
  char *malloc(),*regcmp(),*regex(),*regsub();
  int i,len;

  progname= *argv++;
  if(argc<3) {
    fprintf(stderr,"usage: %s string reg-expr [template]\n",progname);
    exit(0);
  }
  test= *argv++;
  regexpr= *argv++;
  template= *argv++;
  len=strlen(test);
  for(i=0;i<RETNUM;i++)
    if((ret[i]=malloc(len+1))==NULL) {
      fprintf(stderr,"%s: no space for return substring #%d\n",progname,i);
      exit(1);
  }
  if((comp=regcmp(regexpr,NULL))==NULL) {
    fprintf(stderr,"%s: regular expression compilation failed\n",progname);
    exit(2);
  }
  if(regex(comp,test,RETURN_STRINGS)==NULL)
    exit(3);
  else if(strlen(template)==0)
    exit(0);
  else if((result=regsub(template,test,ret,NULL,0))==NULL) {
    fprintf(stderr,"%s: template string substitution failed\n",progname);
    exit(4);
  }
  printf("%s\n",result);
  exit(0);
}
```

and does the '`echo | sed`' task straightforwardly:

`reg` *<font>.<mag-no><type>* `'(.*)$0\.([0-9]*)$1(.*)$2' 'dvifonts/\2/\1/\0.\2'`

There is, unfortunately, a significant change in syntax here. The library regular expression compiler `regcmp()` requires that substrings to be extracted be identified by enclosing them within ordinary brackets `(...)` and suffixing the brackets with `$`*digit*, where *digit* identifies the substring buffer into which the result is to be placed. It's more awkward than the original mechanism, doesn't add any expressive power and, really, just means more complication. Examination of the source suggests that doing it this way made life easier for the implementor of `regex()` ...

Just a little more effort gives you `rel`, which does the '`grep | cut`' or '`cut | grep`' job somewhat more simply:

```
rel -d: -i5 -rpattern -o5 /etc/passwd
```

(The text of the program has been skipped because there's quite a lot of it!) The `-d` flag changes the default field delimiter, as in `cut`, `-i` specifies which input field to key on, `-r` specifies the regular expression to use and `-o` gives the output field specification. The program is considerably smaller and faster-starting than `awk` as well as taking considerably less time to learn to use.

The intention here is not to flash around 'look what a clever program I've just written'—neither of the programs presented here need to be particularly clever, because of the availability of strategically useful and general library routines. No-one in their right mind would voluntarily sit down to write regular expression compiler and execution functions.

Having been deeply immersed in regular expression technology at times over the past year or two, the author now has a mailer whose address transformation mechanism views the world's electronic mail networks as the result of the execution of an often-cascaded system of about 250 regular expression match and replace pairs. Whereas `lex` is designed to handle general forms of input with a specific set of regular expressions, the mailer's address transformation mechanism handles fairly specific input forms with a user-configurable set of regular expressions. The author is fairly convinced that there isn't much that can't be achieved with the correct set of tortuous regular expressions.

## Odds and ends

Thanks Sam.

Alright the rest of you out there? Got that? (This is Doc Strange back again.)

There's just space to mention some odds and ends. First, on my UNIX system administration courses I usually recommend David Fiedler and Bruce Hunter's *UNIX System Administration* (Hayden, 1986) as being the best book there is on the subject. Just recently, though, Rebecca Thomas and Rik Farrow's *UNIX Administration Guide for System V*, Englewood Cliffs, NJ, 1989 (Prentice-Hall, **ISBN**: 0-13-942889-5) has appeared and I think it is even better.

Second, I recently bought a copy of the **JetRoff** shareware Documenter's Workbench postprocessor for HP LaserJet+ and Series II laser printers. If you too only have the binary distribution of Documenter's Workbench and have despaired of getting it to work with your HP laser printer, then I strongly recommend **JetRoff**. It supports all the special characters and drawing commands of the `eqn`, `grap`, `pic` and `tbl` preprocessors, lets you include bitmaps within your document, can print in portrait or landscape orientation, plus much else besides. What's more, as well as the **JetRoff** postprocessor you also get a `troff` preprocessor for making floppy disk and mailing labels, a pretty printer for unformatted text and a wall calendar printer. At $100 (approximately £60) for complete source it is incredibly good value. **JetRoff** is available from: Rick Richardson, PC Research Inc, 94 Apple Orchard Drive, Tinton Falls, NJ 07724, USA, or e-mail to `rick@pcrat.UUCP`.

I have no idea what next issue's column is going to be about. Any suggestions?

# Standards Report on C and POSIX

*Cornelia Boldyreff*
*cs.brunel.ac.uk!corn*

*UK C Panel Convenor*

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

## The Continuing Saga of the C Standard

The ISO C Standard has been undergoing development in parallel with the ANSI C Standard. When this work was proposed by ANSI as a New Work Item to ISO, ISO in approving the work delegated it to ANSI, and ANSI in turn delegated it to the committee within CBEMA, X3J11, which was developing the ANSI C standard. Thus, ISO members participating in the work on the C standard have been reviewing the work of X3J11 and providing X3J11 with critical feedback so that the C standard is acceptable to the international community at large.

A problem has arisen because X3J11 feels they have finished their work, that the draft standard in its present form is 'good enough' [PLA 88]; but some national member bodies within ISO feel that with just a little more effort, it could be better. Two outstanding issues are:

- Readable alternative to trigraphs—a standard solution is required[1]
- Clarity of the standard, in particular, suitability for testing conformance

1. See "A European Representation for ISO C" by Keld Simonsen in this newsletter.

It seems likely that there will be more work on the standard at ISO level to resolve these issues; however, just a little more work is required and we should have an international standard for C that is usable and testable by the end of the year. At the recent meeting of the ISO Working Group on C in April 1989, it was proposed that the following compromise was proposed: to seek international approval of the latest X3J11 draft of the C standard currently undergoing an ISO Draft Proposal ballot and rectify known errors and omissions by means of an Addendum to the standard.

The X3J11 draft standard is still lacking full approval as a national standard by ANSI in the USA. The X3J11 committee neglected to respond to some comments raised during the Public Review and at their recent meeting which coincided with the ISO meeting, they were obliged to put right this oversight.

[PLA 88] P J Plauger, *STANDARD C Addresses Challenge Of The International Market*, The C Users Journal, Dec/Jan 1989.

*Future Meetings*:

| | |
|---|---|
| BSI IST/5/14 C Panel | 7th Feb 1989<br>London, England |
| ISO WG 14 - C | 10-11 Apr 1989<br>Seattle, Washington, USA |

(This will be a joint meeting with X3J11)

| | |
|---|---|
| BSI IST/5/14 C Panel | 9th May 1989<br>London, England |
| BSI IST/5/14 C Panel | 8th Aug 1989<br>London, England |
| ANSI/CBEMA X3J11 | 21-22 Sept 1989<br>Salt Lake City, Utah, USA |

(This meeting coincides with a DECUS meeting.)

| | |
|---|---|
| BSI IST/5/14 C Panel | 7th Nov 1989<br>London, England |

## What's happening on the POSIX Standard Front

The IEEE Standard P1003.1 Portable Operating System Interface for Computer Environments (popularly known as POSIX) has been undergoing a final formal six month ballot at ISO level; if this ballot closes successfully, then the current ISO DIS 9945 will become a full ISO standard. In the UK, the ISO DIS is currently available from the British Standards Institution as a Draft for Public Comment to allow UK experts to have an opportunity to comment on the final draft.

The ISO Working Group on POSIX is anxious to supplement the base standard P1003.1 with the additional POSIX interfaces being standardised by subgroups within the IEEE P1003 committee. To this end, ISO members participating in the POSIX work were balloted regarding subdividing the POSIX work item to accommodate this additional work on the Shell and Utilities, Real-time, Security, Distribution Services, and System Administration as well as the working on various language bindings and a language independent specification of the interfaces. This ballot closed recently and was not unanimously approved; some national members felt that the proposed adoption of the X/Open Portability Guide as a European standard obviated the need for any ISO work related to these supplements. There seems to be something of a misunderstanding here as a European standard cannot hope to address the needs of all the participating international members of the ISO working group on POSIX; the next meeting of the ISO Working Group is scheduled at the beginning of May and I hope to have a more positive report as an outcome of this meeting.

*Future Meetings*:

| | | |
|---|---|---|
| 1-3 May 89 | WG15 - Ottowa, Canada | |
| 19 May 89 | POSIX Panel | BSI, Room 201 |
| 18 August 89 | POSIX Panel | BSI, Room 201 |
| Oct 89 | WG15 - Brussels | |
| 17 November 89 | POSIX Panel | BSI, Room 201 |
| Apr 90 | WG15 - Paris | |
| Oct 90 | WG15 - USA | |

# UNIX Standards: Repertory, Reps and Representation

*Dominic Dunlop*
*domo@sphinx.co.uk*

*The Standard Answer Ltd.*

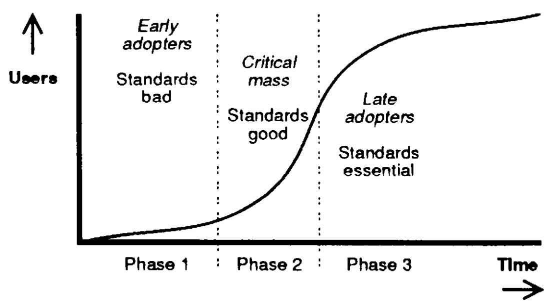
Equipped with an undergraduate degree in Electrical Engineering from the University of Bradford in England, Dominic sidled into the world of mini- and micro-computers. From there, he managed to effect an entry into the hallowed temples of UNIX, and has hung around there ever since, writing the odd paper, contributing to the odd standard, and starting the odd company. He became an independent consultant in January, 1989, and hopes to have the money to buy his latest company, The Standard Answer Ltd., its own UNIX computer Real Soon Now.

There's a small repertory company touring the UNIX conferences of the world this year. I've seen its performances several times, and the players always put on a good show when they are on the stage together—even if the audience sometimes gets a little restless during their monologues.

I'm talking, of course, of the public sparring between the Open Software Foundation and UNIX International over the future of the UNIX operating system. Both organisations welcome invitations to present their points of view on any public platform; those who erect such platforms usually invite both organisations in the interest of balance. Or perhaps in the interest of putting on a good show. When this happened at the recent European UNIX Systems Users' Group conference in Brussels, I was lucky enough to be master of ceremonies at a panel session which followed an afternoon of presentations about UNIX 'Standards'*.

* The quote marks appear in the published programme, as not all the speakers were talking of accepted standards, but rather of things that they wished to see accepted as standards.

While there was some interesting dialogue when everybody was on stage together, most of the conference audience thought most of the afternoon's presentations (the monologues) turgid. Presenters with commercial corners to fight (Bob Duncanson from AT&T UNIX Europe Ltd.; Henning Oldenburg of OSF; Morris Schwartz from UNIX International; and John Totman from X/Open) did not help their respective cases by using standard marketing materials which, although highly polished, had not been prepared with an audience of sceptical hackers† in mind. Walter De Backer of the Commission of European Communities put his finger on the reason why those who attend EUUG conferences dislike sales pitches when he presented this graph (which I have modified somewhat):



In a technology's early days, it is adopted by those who can recognise its merits without needing to be sold the concept. At this stage, standards are a Bad Thing, as experimentation is still required before the technology is ready for wider distribution. Wider distribution comes in the second phase, during which it is possible to begin to formulate standards covering the basic aspects of the technology. The more standards that are formulated, the safer new users feel—and the more susceptible they are to sales pitches. Eventually, in the third phase, even those who have been holding out capitulate and catch up with their rivals by adopting the technology, their old excuse that they are waiting for standardisation no longer being valid. By this time, sales pitches are less necessary: the choice of technology is obvious.

The technology that is UNIX has reached the half-way point of phase two, with phase three looming on the horizon. Consequently, various interest groups are wrangling to make sure that it is their particular view of the world which come to be seen as the obvious answer. (That these views differ only slightly in most respects is masked by the melodrama of the confrontation.) Meanwhile, early adopters who have shown the skill, commitment, and stamina which has been needed to bring UNIX to its current position—and who attend EUUG conferences—see standards as constraining their freedom to experiment. They resent people with suits telling them that previously fluid aspects of their favourite systems have been solidified by decree, or have been outlawed by some standard or another. As one unhappy questioner from the floor put it, ANSI had ruined the C language by defining its every aspect. Where now could he discover esoteric features and work on new techniques?‡

The answer to the general question is that it's always possible to be ahead of the pack, but only if you keep running. While IEEE standard 1003.1-1988 (POSIX) may have nailed down many aspects of the interface to the UNIX kernel, plenty of other areas are still open for research: administration, particularly of distributed systems, is a case in point, so is security, internationalisation... But you'll have to get in quickly, as already there are standards working groups crawling over these and other topics, inventing technology when they come up against an area where there is little experience with practical systems. They may invent something that you don't like, which you could have told them wouldn't work, or which goes against some precedent you could have told them about. You've got to watch these people!

The EUUG is actually paying me to watch the International Standards Organisation on your behalf as Sub-Committee 22, Working Group 15 moves Posix to world standard status. I will be reporting back to you in these articles in the future (while trying not to tread on Cornelia's toes). Please contact me by email if there is any aspect of that standard which concerns you particularly. I may also be able to point you in the right direction if you want to participate in other standards efforts—such as those for administration, distributed systems, and internationalisation. Again, contact me by email. I shall endeavour to give you your money's worth!

† I use the word in its traditional, non-defamatory, sense.

‡ I suggested he should try C++; but perhaps I was wrong: ANSI has started examining that language with a view to its standardisation. Hewlett-Packard and American Airlines have even founded an organisation, the Object Management Group, to look at standardising descriptions of objects across applications.

# Software | Review

*Donal Daly*
*uucp daly@cs.tcd.ie*

*Distributed Systems Group*
*Department of Computer Science*
*Trinity College Dublin*
*Eire*

Donal Daly works as a researcher for the Distributed Systems Group in Trinity. His current work is involved in developing UNIX on top of a object oriented distributed operating system. Previously he had system management responsibilities for System V and Berkeley UNIX systems within Trinity. Donal is the new chairperson of the Irish UUG.

## Introduction

Welcome again to *Software | Review*. The purpose of this column is to review public domain software that appears on the net amongst other places. This offering includes a short review on *perl*. Also reviews on *ream* (a mail reader) and *ssh* (a supershell). These packages have not been posted in the source news groups, but I think they are worth a review, as they are very good examples of the quality of public domain software. Following on from this is a review of a very fast *grep*. Some more detailed information on the upcoming new news software is given. Finally an update to the information I presented about archive sites in the last issue. I hope you enjoy this months offering, and remember I actively encourage you to contribute to this column.

## PERL

*Perl* is a interpreted language optimised for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. Its author is Larry Wall (of *rn* and *patch* 'fame'). Perl is a good language for many system management tasks. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). It combines some of the best features of C, *sed*, *awk*, and *sh*, so people familiar with those languages should have little difficulty with it. Expression syntax corresponds quite closely to C expression syntax. If you have a problem that would ordinarily use *sed* or *awk* or *sh*, but it exceeds their capabilities or must run a little faster, and you don't want to write the silly thing in C, then *perl* may be for you. There are also translators to turn your *sed* and *awk* scripts into *perl* scripts.

*Perl* comes complete with an extensive manual (nearly forty pages). It has a configure script which will work out the system dependencies of your system and edit the *Makefile* and *config.h* as necessary. Compiling and installing *perl* couldn't be easier. There is also a series of test programs provided to ensure *perl* is working correctly. *Perl* also comes with a number of examples (I have a number of other ones which I can send to people also). The way *perl* is packaged should serve as a good example in how all major public domain packages should be packaged before being released. Larry has also written a configure generator (*dist2* in *comp.sources.unix* volume 16) to help software developers create these magical configure scripts.

From recent email I had with Larry, here are some of the features to expect in the next release of *perl*: (apart from little bug fixes) the support for arrays and array handling has been greatly improved. You can now do arbitrary *ioctl* functions. Other new functions include: *mkdir*, *rmdir*, *getppid*, *getpgrp*, *setpgrp*, *flock*, *rindex*. Error reporting and syntax checking is improved. There should also be socket and select support for those systems that can support them. *Perl* can now perform greater optimisation on *perl* scripts.

The current release is 2.0 patchlevel 10. *Perl* was published in *comp.sources.unix* volume 15. I found perl to be a very useful tool and look forward to its next release. I would recommend it as the language of choice for fast prototype or for simple system management tasks.

## SSH

**(Paul Dorish jpd@etive.ed.ac.uk)**

Like most modern shells, ssh features filename completion and command-line editing. The editing, however, is completely configurable—you can either set it up on the fly, or use one of the predefined keysets (the distribution version includes modes for *em*, *vi*, *emacs* and *tcsh* emulation).

History is also here, and job control too. However, ssh goes further than most job control shells, by offering job control on System V! Sxt devices are used to block I/O, giving pseudo job control in much the same way as *shl*, but much more conveniently, since it uses the familiar BSD *csh* system (suspend characters, *fg*, *bg*, *jobs*, etc).

In fact, this System V job control facility is a clue to another important *ssh* feature, which is wide portability. It originally ran under Version 7 (and still does)—but since then it's been ported to more machines than most of us will encounter, under variants of System V, BSD 4.x (and 2.x), Version 7 and a few rather strange 'homebrews'.

Now, what about those really strange features I mentioned? Well, how about system calls available from shell script? Yup, I've seen network daemons listening on IP sockets and answering requests written as *ssh* script. You can also redirect to sockets, if you have the right facilities (or FIFOs, if you prefer). And for something really strange, how about a DWIM ("Do What I Mean") facility? Imagine:

```
$ mroe file
not found: "mroe" -> more
```

I didn't make that up—I included it from an *ssh* window! Before you worry about it translating 'mr' into 'rm' accidentally, I should add that the cursor sits after the correction, waiting for you to accept, reject or edit it.

If GNU Emacs were a shell, it would be *ssh*. And look how popular Emacs is! [Actually, Simon will hate me saying this, being a confirmed *vi* hacker; but if he can't take a compliment...]

*Ssh* is available in source form from Simon Brown. Try mailing him at *simon@meiko.co.uk*.

## REAM

**(Simon Brown - simon@meiko.co.uk)**

Think of how it is when new mail is arriving all the time, but you don't read it because it's just too much effort. Or reading your mail takes up all your free time, so you're threatening to just give up on it. Well, it's not the mail itself that's the problem—it's the mail reader you're using. "But who wants yet another mail reader?"

This is where Ream (Paul Dourish's e-mail reader) comes in. Ream is to other mailers what 'vn' is to 'readnews' for reading Usenet news! It provides an environment specially designed for dealing with mail in the most flexible way possible.

Ream presents the user with a screen of mail header information, allowing quick and easy access to all of your mail simultaneously. It provides a very sophisticated message naming scheme to allow any mail message, or group of messages, to be dealt with in any of a vast number of ways. This is particularly useful if you receive a large amount of mail—you can see at a glance what mail you've got, and deal with it accordingly.

The most common problem with mail reading tools is that they hide too much detail of the mail mechanisms, causing unbearable frustration for those dedicated mail enthusiasts who love nothing better than looking through pages of routing information. But the other extreme is equally unpleasant—nobody can use /bin/mail as a mail reader for very long and still keep a remnant of sanity. What is needed is a highly flexible tool that can be tailored exactly to an individual's needs, but which can still be used in a raw default form by novices.

So Ream is more than a simple mail reader—it provides an entire environment dedicated to the efficient and easy manipulation of mail. Mail folders can be loaded in or written out, just as an editor deals with files. Pattern matching is available, whereby messages may be selected by any of a number of attributes. You want to save all messages coming from a particular person into a particular file? Easy—a few keystrokes will do the trick. You want to format the header fields of mail messages you send out, in a particular style of your own? Easy—a few lines in your personal configuration file will do it for you. You want to live in safety and be sure of never losing anything by typing the wrong thing by mistake? Easy—Ream can look after you and query commands that seem dangerous, if you want this. If you don't, then of course you don't get it.

This is the whole philosophy of Ream—it's all environments to all people. Nothing is hard-wired, everything is programmable. But the defaults (or should I say the 'default defaults'?) are arranged so as to make it usable from the very start.

Ream has been widely ported—it runs under both Berkeley and System V UNIX, on a vast variety of hardware, and can be used with either Sendmail or MMDF (the two most common UNIX mail delivery agents).

Ream is available in source form from Paul Dourish at the University of Edinburgh Computing Service, where Ream is now the standard supported mail tool. Try mailing him for details—his email mail address is *jpd@etive.ed.ac.uk*. You can be sure of a prompt reply—after all, although he receives more mail per day than most people see in a month, he has the most sophisticated mail-reading environment you could hope for at his disposal...

## Woods/Spencer Grep

**(Mike Selway mas@ssl.uucp)**

Everybody who is really in the know, knows that egrep is the fastest of the three greps. The real scientists among us know that egrep is faster because they've actually timed it. And the figures are impressive: for a simple dictionary search on my machine, taking grep as 100%, fgrep takes 82% and egrep takes 42%. That saves a lot of time if you're searching through megabytes of disc space.

Compare this with wc at 41% and it looks like egrep is close to optimum.

I had been aware of various greps which had emerged in recent times which were supposed to be quite fast: Andrew Hume's paper at the London EUUG conference described work he had been doing to speed things up. However, when someone sent me a copy of the Woods/Spencer grep from comp.sources.unix I wasn't really prepared for the for the results I got.

The simple search which egrep cuts to 42% is cut to just 17%. On one of our development machines this grep runs at the same speed as cat if the pattern matches every line. The inner loop of this program compiles to just 4 instructions on a 68020, and it runs faster on a Sun 3 than standard egrep on a Cray 2.

Woods/Spencer grep has some very fast code for spotting exact matches which it uses to determine whether the line is worth presenting to the regular expression code. If the expression is very complex or you invoke some obscure options, or, indeed, if it notices that it's taking a long time to get through the first bit of the file, grep will just exec the most suitable standard version from /bin or /usr/bin. In this way it limits its worst case behaviour. Unfortunately, if your standard grep family takes a different set of flags from those that the new grep expects it to take, you can get some odd error messages:

```
$ newgrep -i 'a*b*c*d' /usr/dict/words
newgrep: illegal option -- i
 usage: egrep [ -bclnv ] ...
```

A GNU grep is now available which I'm told is a later version of the Woods/Spencer grep, but without the exec feature.

Woods/Spencer grep is available for UKNET sites in the UK from the *comp.sources.unix* archive on the Kent info-server as *volume9/fastgrep/part01* and *volume9/fastgrep/part02*.

## NewsFlash

Here is some more detailed information on the upcoming new news software (really I just wanted to use that heading again :-)). At the time of writing neither of the news software packages has been released. The overviews of the new news software are by the authors themselves.

## C News

**(Geoff Collyer geoff@utstat.toronto.edu)**

'C News' is a rewrite of the transport and expiry subsystems of the Usenet news software. It is approximately compatible with B News 2.11, which is the current production version in most places. In particular, it preserves full compatibility in the way articles are stored and in the format of control files, so that existing news-reading programs work with it. C News doesn't include any new news readers, in fact: it has the Australian reimplementation of 'readnews' for naive users, and 'rn' for sophisticated ones. Various other news readers have been tried with it; they all work.

The main benefits of C News are performance and robustness. It was written to try to solve some of B News's performance problems. It's fairly successful at that; the current C News processes incoming news at roughly 25 times the speed of B 2.11. On a fast machine like a Sun 3, C News's performance approaches the theoretical ultimate, since a large fraction of its time is spent in system calls, and considerable pains have been taken to minimise the number of calls needed.

C News offers some other minor benefits. It uses shell files wherever possible, which makes maintenance simpler. Its 'expire' runs from a control file which allows specifying things like expiry times on a per-newsgroup basis. The code is generally fairly cleanly written, unlike the unholy mess inside 2.11.

At the moment, the main disadvantage of C News is that it is in a private beta test. An 'alpha' release was distributed via *comp.sources.unix* a while ago, with some minor updates sent out in *news.software.b* later. The alpha release runs fairly well, but it has a number of annoying quirks and minor problems and is *very* poorly documented. A definitive release has been 'in the works' for some time, and we hope to actually get it out the door this spring.

C News was written primarily by Geoff Collyer and Henry Spencer at the University of Toronto. It is unrelated to B News 3.0, a separate project which is also underway right now.

## News 3.0

**(Eric S. Raymond eric@snark.uu.net)**

The software that will be 3.0 netnews is in final beta at sixty sites now, including one in England. It is a complete rewrite of the news suite, including both an improved transport layer, a complete suite of readers, tools and service libraries for building custom interfaces, and extensive documentation. The code (which has been reorganised as a stack of abstract data types in a way somewhat analogous to the ISO 7-layer networking model) is 8-bit clean.

Notable new features include: full support for conversation-following in the readers (using the followup-to relation and its inverse); a customisable summary-page mode that assists in filtering a lot of news rapidly; rn-style kill-language processing in every reader; a '-' command that really works (one can back up clear to the beginning of a session along the conversation track, then forward again).

Also, setup and administration have been greatly simplified. Much closer control of article expiration is supported. Support for multicasting and exotic point-to-point layers is improved. Extensions to the subscription syntax make specification of complex partial feeds less cumbersome.

Though completely functional in itself and upward-compatible with older news versions, the rewrite is also the first step in a plan for implementing planetary-scale distributed hypertext using the netnews protocols as a base. For further information and access to beta sources, contact the author, Eric S. Raymond, at *eric@snark.uu.net*.

## Archive Servers

Quite a few of the responses I have had to my first article went something like this ''Can you send me this public domain software'' or ''How can I get hold of these public domain sources?''.

Firstly a lot of this public domain software is available in the form of EUUG tape distributions. Contact Frank Kuiper (*euug-tapes@cwi.nl*) for more information. The latest conference tape from the Brussels conference contains the most recent versions of the GNU sources.

The EUnet backbone managers are currently investigating how a EUnet wide archive service can be set up. This would be available to you via a simple email message to your local backbone site. I will publish more information when details have been finalised. For now, here are details of two backbone sites offering an archive site. Thanks to Keld Simonsen (*keld@dkuug.dk*) and Peter Houlder (*uknet@ukc.ac.uk*) for the information.

### Denmark

In Denmark there is a mail-based news and source archive service reachable at *archive@dkuug.dk*. Mail it with Subject: help for further information. The archive contains news and selected portions of the latest EUUG tapes (GNU and X excluded). It is restricted to use within Denmark for accounting reasons.

### England

In England there is a mail-based source archive service reachable at *info-server@ukc.ac.uk*. This server contains several public domain system including the news code and *comp.sources.unix*. There is also a server at *netdir@ukc.ac.uk* which will return you the uucp map information for a site specified in the subject line of the message. For more information about UKnet archive services see the last EUnet column.

## Next Issue

As ever, I am interested in comments if you found anything useful or interesting. Remember I actively encourage you to contribute to keeping this column alive, so email me your reviews of the latest and greatest piece of software you use. I would really like to get some reviews on any of the GNU software. I will gather them together and publish them in a special section on GNU, hopefully in the next issue.

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

## Collective noun phrases from the computer field:

a batch of mainframes
a deck of VAXes™
a clique of workstations
a double clique of Macintoshes™
a session of terminals
a stack of programmes
a heap of programmers
a class of object-oriented programmers
a relationship of database designers

The list can certainly be augmented and improved!

Best regards,

Markku

Markku Sakkinen
I markku@jytko.jyu.fi
Department of Computer Science, University of Jyväskylä
Seminaarinkatu 15, SF-40100 Jyväskylä, Finland

# USENIX Association News for EUUG Members

*Donnalyn Frey*
*donnalyn@frey.com*

*Frey Communications*

Ms. Frey is the USENIX Association Press Liaison. She provides members of the press, USENIX Association members, and EUUG members with information on the activities of the USENIX Association.

## 1989 Summer USENIX Association Conference and Exhibition

The USENIX Association's 1989 Summer Conference and Exhibition, held June 12-16 in Baltimore, Maryland, was a success. Over 3,500 people attended the conference. The first two days were devoted to tutorials, with the next three days for technical sessions. The technical exhibition was held on June 13, 14 and 15. Bill Wulf, of the National Science Foundation, presented the Keynote Address entitled "What Will We Do With All Those Cycles In The Year 2000?".

Tutorials were presented on many subjects, including MACH, 4BSD TCP/IP performance improvements, internals of the GNU C compiler, security issues, Postscript, AIX technology, System V internals, network programming, C++, software contracts, and programming in the X Window System.

Technical papers were presented on streams and the kernel, networks, administration, windowing systems, program development, file systems, security and performance issue. A Work in Progress session was held on Thursday afternoon to preview new work not yet ready for publication.

### Security

Two full sessions were held on computer security, including a paper entitled "Who Can You Trust?" by Thomas Malarkey of the National Computer Security Center. Other papers discussed controlling gateways, security testing, and B1 security portability.

### Terminal Room

The USENIX Association again sponsored a terminal room at the Baltimore Conference. Attendees could read their mail or contact their offices from terminals at the conference.

### FaceSaver

The FaceSaver returned to the USENIX Association Conference. The FaceSaver was operating in the vendor exhibition area, with on-line access to faces on uunet. Attendees were able to have their faces saved and uploaded to uunet.

To request copies of the Summer 1989 USENIX Association Conference Proceedings, send email to {uunet,ucbvax}!usenix!office or office@usenix.org.

## 1990 Winter USENIX Conference

The 1990 Winter USENIX Conference will be held in sunny Washington, DC on January 20-26, 1990. The first two days will be devoted to tutorials, with the next three days for technical sessions. For further information on the conference, contact the USENIX conference office.

## Distributed Processing Workshop and Graphics Workshop

The USENIX Association will be holding a Distributed Processing Workshop in Fort Lauderdale, Florida October 5-6, 1989. The fifth Graphics Workshop will be held November 16-17,1989 at the Doubletree Hotel in Monterey, California. For information on these workshops, contact the USENIX Association Conference Office at 22672 Lambert Street, Suite 613, El Toro, CA 92630, USA.

## Further Information on Conferences and Workshops

If you need further information on upcoming annual USENIX Association conferences or workshops, contact the USENIX conference office at its new location at 22672 Lambert Street, Suite 613, El Toro, CA 92630, USA. The conference office can provide you with information on the annual Computer Graphics, Large Installation Systems Administration, UNIX Security, and UNIX and Supercomputers workshops, as well as the new workshops on Software Management and Transaction Processing. The office can also provide information on the 1990 C++ conference and the semi-annual technical conferences. A schedule of upcoming events includes:

### Postscript

The USENIX Association regrets to announce the departure of Dr. Peter Salus, outgoing Executive Director. Dr. Salus has taken a position as Director of University Relations at the Open Software Foundation. The Association is pleased to announce the appointment of Ellie Young as the new Executive Director. Ms. Young has been the Assistant Executive director for much of 1988 and into 1989. The USENIX Association is looking forward to a rewarding working relationship with Ms. Young.

The USENIX Association also announces that it is changing its mailing address. The actual office is not moving. However, the new mailing address is:

USENIX Association
2560 Ninth Street
Suite 215
Berkeley
CA 94710

The telephone number: +1 415 528 8649 will not change.

The electronic mail address remains as:

*office@usenix.org*

USENIX Association Future Meetings

| Date | Location | Topic |
|---|---|---|
| 1-2 May 1989 | Pittsburgh, PA | Transaction Processing Workshop |
| 12-16 June 1989 | Baltimore, MD | Semi-Annual Conference & Exhibition |
| 7-9/8 September 1989 | Austin, TX | System Administration III Workshop |
| 5-6 October 1989 | Ft.Lauderdale, FL | Distributed Systems Workshop |
| 16-17 November 1989 | Monterey, CA | Graphics Workshop V |
| 23-26 January 1990 | Washington, DC | Semi-Annual Conference |
| 11-15 June 1990 | Anaheim, CA | Semi-Annual Conference and Exhibition |
| 22-25 January 1991 | Dallas, TX | Semi-Annual Conference |
| 10-14 June 1991 | Nashville, TN | Semi-Annual Conference & Exhibition |
| 20-24 January 1992 | San Francisco,CA | Semi-Annual Conference |
| 8-12 June 1992 | San Antonio, TX | Semi-Annual Conference & Exhibition |

EUUG

European UNIX® systems User Group

# 4.3BSD MANUALS

The USENIX Association now kindly offers all members of the EUUG the opportunity to purchase 4.3BSD Manuals.

The 4.3BSD manual sets are significantly different from the 4.2BSD edition. Changes include many additional documents, better quality of reproduction, as well as a new and extensive index. All manuals are printed in a photo-reduced 6″ x 9″ format with individually coloured and labelled plastic 'GBC' bindings. All documents and manual pages have been freshly typeset and all manuals have 'bleed tabs' and page headers and numbers to aid in the location of individual documents and manual sections.

A new Master Index has been created. It contains cross-references to all documents within the other six volumes. The index was prepared with the aid of an 'intelligent' automated indexing program from Thinking Machines Corp. along with considerable human intervention from Mark Seiden. Key words, phrases and concepts are referenced by abbreviated document name and page number.

While two of the manual sets contain three volumes, you may order complete sets only.

The manuals are available now. To order return a completed '4.3BSD Manual Reproduction Authorisation and Order Form' to the EUUG secretariat along with your remittance. You must be an EUUG member.

The EUUG has bulk shipped these manuals from the USA thereby saving you 5 kg transatlantic postage.

| Manual | Cost |
|---|---|
| **User's Manual Set (3 Volumes)**<br>**User's Reference Manual**<br>**User's Supplementary Documents**<br>**Master Index** | **£25.00/set** |
| **Programmer's Manual Set (3 Volumes)**<br>**Programmer's Reference Manual**<br>**Programmer's Supplementary Documents, Volume 1**<br>**Programmer's Supplementary Documents, Volume 2** | **£25.00/set** |
| **System Manager's Manual (1 Volume)** | **£10.00** |

*4.2BSD Manuals are No Longer Available*

## 4.3BSD Manual Reproduction Authorisation

Date ____________

As an EUUG member† in good standing, and pursuant to the copyright notice as found on the rear of the cover page of the Unix ®/32V Programmer's Manual stating that:

"Holders of a Unix ®/32V software licence are permitted to copy this document, or any portion of it, as necessary for licenced use of the software, provided this copyright notice and statement of permission are included."

I hereby appoint the USENIX Association/EUUG as my agents, to act on my behalf to duplicate and provide me with such copies of the Berkeley 4.3BSD Manuals as I may request.

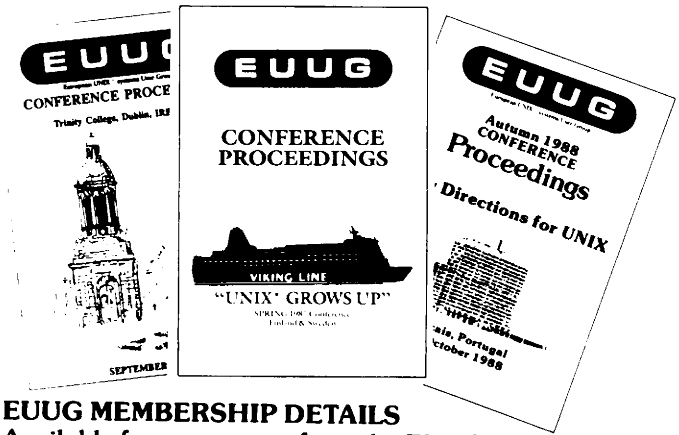
Signed ____________________________

Institution (if Institutional Member) ____________

† Members of EUUG National groups are automatically members of the EUUG.

# EUUG

European UNIX® systems User Group



## ADDITIONAL PUBLICATIONS AVAILABLE FROM EUUG

| Title | | Cost |
|---|---|---|
| **EUUG – CONFERENCE – PROCEEDINGS** | | |
| Spring 1983 | – Bonn | £2.00 |
| Autumn 1983 | – Dublin | £2.00 |
| Spring 1984 | – Nijmegen | £5.00 |
| Autumn 1984 | – Cambridge | £5.00 |
| Spring 1985 | – Paris | £5.00 |
| Autumn 1985 | – Copenhagen | £10.00 |
| Spring 1986 | – Florence | £20.00 |
| Spring 1987 | – Finland/Sweden | £20.00 |
| Spring 1988 | – London | £20.00 |
| Autumn 1988 | – Portugal | £20.00 |
| European E-Mail Directory | | £18.00 |
| 4.3BSD MANUAL (Members only) | – User's Manual Set | £25.00 |
| | Programmer's Set | £25.00 |
| | System Manager's Manual | £10.00 |
| Language C – Standard Proposal | | £8.00 |
| USENIX Conference Proceedings C++ – Santa Fe, 1987 | | £30.00 |

**EUUG MEMBERSHIP DETAILS**
Available free on request from the EUUG Secretariat.

**UKUUG PROCEEDINGS**
Winter 1988 –UKNet Meeting, Canterbury **£10.00**

If you wish to order any of the above publications, or receive details of EUUG Membership, please use the order form below and return it to the EUUG Secretariat, Owles Hall, Owles Lane, Buntingford, Herts. SG9 9PL, U.K. If you are ordering publications then your remittance should be enclosed by cheque, bank draft etc. made payable to EUUG.

## ORDER FORM

Name ____________________

Address ____________________

____________________

____________________

____________________

Phone ____________________

Network Address ____________________

The prices shown include surface postal charges. All payments must be in Sterling by means of a cheque drawn on a UK bank, or a Eurocheque, or instruction to charge your VISA card (quoting the number, the date of expiry, the name of the card holder, and the address which VISA use when corresponding with you).

Please send me ______ no. copy/ies of the following:

____________________

____________________

I would like to receive membership details of the EUUG YES/NO

| | |
|---|---|
| User's Manual Set (3 Volumes) | at £25.00/set = £ ______ |
| Programmer's Manual Set (3 volumes) | at £25.00/set = £ ______ |
| System Manager's Manual (1 volume) | at £10.00/set = £ ______ |
| Total | £ ______ |

We accept
Visa, Access, Euro Card, Master Card, UK cheques, and Euro cheques.

Please complete name and address of credit card holder if different from name and address on order form.

NAME ____________________

ADDRESS ____________________

____________________

____________________

Expiry Date: ____________________

Credit Card No:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

I enclose my cheque, bank draft etc., in the sum of £ ______ and understand a receipt will be sent.

Make your cheque payable to 'EUUG' and post it with this order form

To: **EUUG Secretariat, Owles Hall, Owles Lane, Buntingford, Herts. SG9 9PL, United Kingdom.**
**Tel: Royston (0763) 73039 +44 763 73039**
**Fax: Royston (0763) 73255 +44 763 73255**

For BSD manuals please sign Manual Reproduction Authorisation Notice overleaf.

This page may be photocopied for use.

# UNIX System V Data Networking Architecture

*Laurence M Brown*

*AT&T*

Laurence Brown is a supervisor in the Network Systems Engineering and Architecture Department at AT&T in Summit, New Jersey. He has worked on the design of operating systems, computer networks and database management systems and was responsible for the architecture of the networking features of the UNIX System V Release 3. He is currently responsible for the architecture of full OSI protocol stacks using the STREAMS feature of UNIX System V Release 3. He holds an M.S in Computer Science from Stanford University.

please contact Janet Davis <janet@uel.uucp> for more information on the contents of this column.

The computer industry is converging on two communications standards—Open System Interconnection (OSI) and Systems Networking Architecture (SNA). OSI will be used in multivendor environments such as factory automation, office automation and telecommunications. OSI is at the heart of the AT&T Data Networking Architecture. SNA will be used to communicate with corporate mainframes UNIX systems must be able to communicate fluently using both of these standards and must provide a smooth migration path to these standards.

The STREAMS facility of UNIX System V Release 3 provides a powerful tool for handling these problems. This paper describes the properties of the STREAMS technology that makes it ideal for the implementation of communication protocols. The STREAMS facility of UNIX System V Release 3 provides a flexible and efficient mechanism for solving a variety of communications problems.[123] This article describes how the STREAMS facility can be used to solve challenging networking problems. The first section of this article examines trends in networking that present both challenges and opportunities for UNIX System vendors. The second section identifies the attributes that the preeminent networking operating system. The third describes how the STREAMS technology can be exploited to provide these attributes. The final section shows how popular networking architecture can be implemented in a consistent and compatible fashion using the STREAMS facilities.

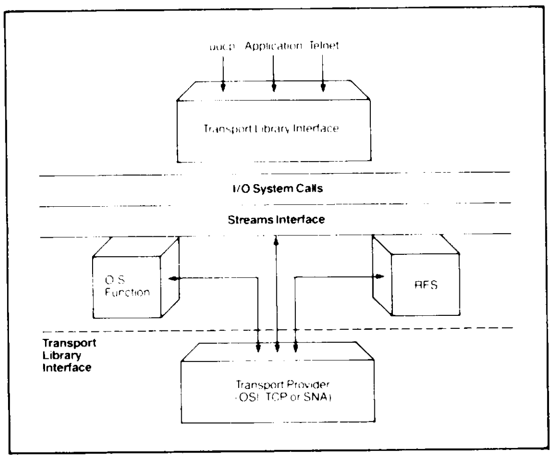

uucp Application Telnet
Transport Library Interface
I/O System Calls
Streams Interface
O S Function
RFS
Transport Library Interface
Transport Provider (OSI TCP or SNA)

**Figure 1.** Transport Layer Interface
The Transport Library Interface is a shared library, used by applications and standard networking utilities. These applications and services are made protocol-independent through TLI. Any transport provider (OSI, TCP or SNA) can be used to provide the transport service.

## Trends in Networking

The successful computer vendor today must be able to cope with a wide variety of communications standards. In broad terms, the communications standard required is determined by the target market. In the scientific, academic, defence and computer-aided design (CAD) markets, the Transmission Control Protocol/Internet Protocol (TCP/IP) defined by the U.S Department of Defence and popularised by the ARPANET (Advanced Research Project Administration Research Network) and 4.2 BSD (Berkeley System Distribution—a derivative of the UNIX operating system) are predominant. Office Automation is currently an unstable market in terms of communications standards, but products based on XNS for Xerox have the largest current market share. Factory automation requires the OSI protocols as specified in the Manufacturing Automation Protocol (MAP) standards. IBM's Systems Network Architecture (SNA) has the major position in the Back Office Market. The successful vendors will be the ones who can handle diverse standards with consistency and provide smooth migration to a more orderly future.

In the future, OSI and SNA standards will survive at the most prominent. Each will have its own sphere of operation and the successful vendors will be the ones who can provide interworking between them.

### OSI

OSI protocols will be the standards of choice in any open market. AT&T's Data Networking Architecture is based on OSI. Included in this category are markets that are inherently multi-vendor such as Office and Factory Automation and markets where government policy preclude single vendor dominance.

The OSI protocols are a good choice for these markets since the OSI protocols are open standards that have been designed and evolved through a consensus the last five years, the OSI protocols now represent a complete solution to current networking problems and provide a solid architecture on which to evolve future solutions. Thanks to the efforts of the Corporation for Open Systems (COS) in the United States and the Standards Promulgation and Acceptance Group (SPAG) in Europe, there are implementable subsets of OSI that customers are demanding from the vendors.

### SNA

SNA is popular in the Back Office because of the large investment in IBM architecture machines that speak SNA. Any vendor who wishes access to the valuable information stored on these corporates mainframes must be prepared to speak SNA.

## Challenges for UNIX System V

To deal with the evolution of these networking standards, UNIX System V must be very nimble in its support of networking. The basic challenge is to support all of these different protocol suites in a consistent manner that will allow vendors to support multiple protocol standards and allow customers to migrate from one to another without rewriting their applications. This presents specific challenges for UNIX System V.

### Application Portability

The first challenge for System V is to ensure the same degree of portability for networking applications that the System V Interface Definition guarantees for traditional applications. For networking applications it is important to provide portability across protocol suites as well as across machines. The need is for a standard interface to networking services that is independent of the protocol that provides the service. This means that an application written to this interface must work whether the underlying network is SNA or OSI or TCP/IP or XNS. In order to have a precise and unbiased model of networking, this interface should be based on the service definitions of the OSI reference model.[4]

### Protocol-Independent Networking Services

The second challenge involves providing the useful networking services that customers need in a protocol-independent manner. Included in this set are traditional networking services such as electronic mail, file transfer, and remote execution, as well as industry specific services such as videotex and electronic funds transfer, and also traditional operating system services such as the file system or process management distributed across the network. Traditionally these services have been constrained to work over a single set of protocols. However, if UNIX System V is to be an open communication environment, it must provide a uniform set of networking services that work in any protocol environment.

### Protocol Implementation Environment

The third challenge is to meet the needs of the protocol implementors. The operating system must provide the muscle and sinew necessary to implement modern protocol architectures in a modular and reconfigurable fashion.

**Protocol Substitution.** The first requirement is to support protocol substitution—that is the ability to dynamically substitute one protocol, say TCP, for another. functionally equivalent protocol, say ISO Class 4 Transport. This is the facility that is needed to support a protocol independent application network services. In addition, this facility is needed if a vendor needs to use a single protocol, such as X.25, in several different protocol architectures, such as SNA, TCP/IP and OSI—all of which require X.25 wide area networking.

**Protocol Portability.** The second requirement is that the operating system should provide an environment that allows protocol modules to be portable in the same way that applications are portable today. This hasn't been the case previously. For reasons of efficiency and timing, protocols have been implemented in the UNIX System kernel, but kernel code is not typically portable. Any manufacturer who has a wide range of machines would benefit from being able to rapidly part a single communications package across the entire line. In addition, portable protocols could be useful to the computer industry as a whole. An inordinate amount of time is spent in verifying protocol correctness and multivendor interoperability. If there were standard reference implementations of the protocols that any vendor could port, the need for extensive multivendor testing would be reduced.

**Internetworking.** The nature of modern communication architectures present the third requirement. To support internetworking, the kernel must facilitate the routing of data to multiple device drivers. The traditional UNIX System I/O architecture has a linear connection between the file descriptor and the device driver. Protocol developers have had to use tricks to program around this constraint. The operating system should make this task easier.

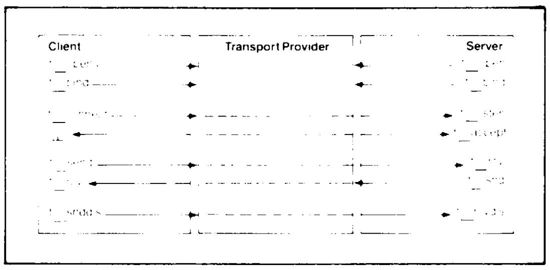


**Figure 2.** Client /Server Interaction Using TLI
The client who makes a call and the server who responds use a set of primitives to talk to each other. Both client and server must open the Transport Provider and bind to a local address. The client issues a connect request, and the server listens for incoming connect requests on that address. Once the virtual circuit is established, either side may send or receive data until a disconnect occurs.

## Meeting the Challenges with STREAMS

How STREAMS technology can be used to meet these challenges will be discussed in the following paragraphs. STREAMS is designed to support a multitude of protocols of diverse functionality and to provide broad support for networking and internetworking systems.

The STREAMS framework does not impose any specific network architecture but enables various protocols to be easily implemented. Further, it facilitates the development of applications independent of the underlying protocols.

### Common Application Interface

The level at which a common application interface can be defined depends on the characteristics of the protocol suites that must be supported. Given the currently popular protocol suites—SNA, TCP/IP, OSI and XNS—this interface must be at the Transport Layer in the OSI reference model. Transport protocols provide reliable machine to machine communication independent of the underlying network topology. In a sense, this is the fundamental networking service. It is the foundation for all higher level services. It would be undesirable for an application or a file transfer protocol to have to worry about the details of error detection and recovery or routing. Since Transport is such a valuable service, every major protocol family has a Transport protocol. In XNS it's SPP. In TCP/IP it's TCP. In SNA it is provided by the basic conversational verbs of LU6.2. In OSI there are five different transport service, but they differ in the assumptions that they make about the underlying network layer protocols.

The strategy in System V Release 3 is to provide an application interface to Transport Services that is based on the formal OSI definition of Transport Services.[5]

There are three types of OSI documents. The seven layer OSI Reference Model gives the general framework for modern networking architectures. It defines terms and partitions networking functionality into distinct layers. It provides the tools necessary to meaningfully compare protocol architectures. However, it does not precisely define the services provided by each layer. The second OSI document type, the OSI Service Definitions, defines in precise terms the obligations of a protocol at each layer of the Reference Model. There is a potentially infinite set of protocols that can provide these services, but the consumer of the services has an unambiguous definition of the services that any of these protocols would provide. The final level of OSI documentation is the Protocol Specifications themselves. A Protocol Specification defines the operation of a particular protocol that meets an OSI service definition. The strategy in UNIX System V is to base the application interfaces for networking services on the OSI Service Definitions. These interfaces are defined as 'C' language library interfaces through which an application can request networking services at the appropriate layers in the reference model. The customer can then position under this interface any protocol that provides the desired service.

The UNIX System V Release 3 library for Transport Services is called the Transport Library Interface (TLI). TLI is a UNIX System user level shared library through which an application can request reliable data communication. This interface has been engineered so that it will work with all the popular protocol suites. The result is that an application developer or the developer of a networking service can write a program and be guaranteed that it will work without modification over SNA or OSI or TCP or XNS or any other provider of reliable data communications. TLI provides the foundation for protocol-independent applications in UNIX System V. It is specified as a standard interface to networking services in the System V Interface Definition (SVID)[6] and, as such, is included in the System V Application Operating Environment (AOE).

TLI provides an essential service that will be used by application developers for many years to develop applications that will work in a wide variety of networking environments. However, the time will come when it will be commercially important only to write applications that will run in the preeminent protocol environments—OSI and SNA. At this time, it will be appropriate to move up in the Reference Model to the application layer—specifically to the ISO TPE /SNA LU 6.2 level. OSI and SNA have a common service interface to Transaction Processing services at this level. In AT&T, we are currently defining a UNIX System Library interface that will give an application access to Transaction Processing services in both SNA and OSI. Using STREAMS, the appropriate protocol suite can be substituted under the interface to communicate with the customer's network.

A detailed look at TLI will show how these Service Interfaces work. TLI provides three groups of services. The first is 'Common Management Primitives' that are used to bring the application into communication with the local Transport Provider. Through this interface the application can open the Transport Provider, bind itself to a local address, and negotiate any protocol options or discover any boundary conditions.

The second group of services is 'Connection-Mode Transport Services'. These services are only provided by protocols that support reliable virtual circuit service. Included here are TCP, SPP (from XNS), the five classes of OSI Transport Protocols and LU 6.2 (from SNA). Through these primitives, an application can open a virtual circuit, send and receive data from the circuit, and release the virtual circuit.

The third group of services is 'Connectionless-Mode Transport Service Primitives'. These are unreliable datagram services that are provided by UDP from the DARPA protocols and the OSI Connectionless Transport protocol.

Figure 1 shows how the Transport Library interface (TLI) looks in the UNIX system. TLI is a shared library. An application, or a network service such as uucp or telnet, can receive Transport Services by making calls to the library interface routines. The TLI Library communicates these services requests to the Transport Provider by making STREAMS system calls. The Library uses getmsg and putmsg to send requests to the provider and to receive results from the provider. The provider sees these requests as STREAMS messages. Using the dynamic nature of STREAMS. any provider can be plugged under the service interface to provide the Transport Service.

The messages of the Transport Provider Interface (TPI) can also be generated directly by an object in the kernel sitting upstream from the Transport Provider. The UNIX System V Release 3 Remote File Sharing facility uses this capability to achieve its protocol independence.[7] This same technique could be used by a higher level protocol in the kernel to become transport independent.

Figure 2 shows the actual primitives that two applications would use to talk to each other over an arbitrary network using reliable virtual circuits. There are two parties: the client who makes the call and the server who responds.

The client first opens the Transport Provider and binds itself to a local address on that provider. This is the common management phase. Once bound, the client can issue a connect request. When the connect request returns, a virtual circuit has been established. The client can send and receive data over the virtual circuit. The conversation is terminated when either side sends a disconnect request. All of these primitives are part of Connection-mode Transport Service.

The server must also first open the Transport Provider and bind itself to a local address on the Transport Provider. It is this address that the client gives in the connect request. Once the server has bound itself to an address it can then listen for incoming connection requests on that address. When the server accepts a request, the connect request on the client side returns and the virtual circuit is established. Either side may now send or receive data over the connection until the disconnect occurs.

## Standard Services

A standard set of protocol-independent end-user networking services can be provided by implementing the services to TLI. Figure 1 also shows three services that have been made protocol independent through TLI. UUCP is the standard UNIX system utility for file transfer, terminal emulation, remote execution and mail. Previously, it was limited to operation over dial-up lines. Through the auspices of TLI it works over any valid Transport Provider in System V Release 3. Telnet is the standard DARPA service for terminal emulation. Traditionally it has worked only over TCP/IP, but through TLI it can work over any transport protocol. The same has been done for the other DARPA services -FTP for file transfer and SMTP for mail. Since RFS has been implemented to the kernel TPI, customers can interactively share files over any of the commercially important protocol suites. All the networking services in the AOE are protocol-independent since they have been implemented to TLI.

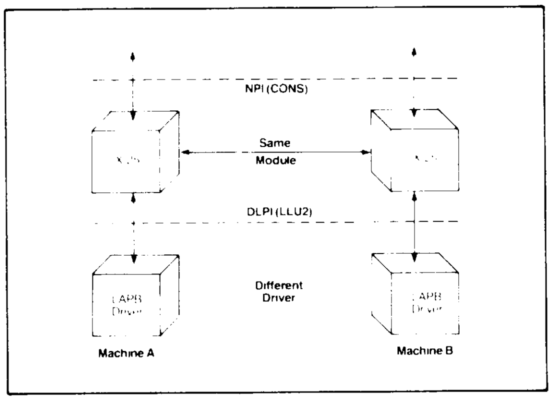


**Figure 3.** Protocol Portability
A protocol implemented in STREAMS is portable to any machine that supports the STREAMS environment and has a STREAMS provider that provides the services that a protocol requires. In this figure, an X.25 module has been ported to Machine B.

## Protocol Substitution

Protocol Substitution is a straightforward application of the service interface concept. Figure 1 shows an operating system function, for example, process management, that is being implemented to work over a network. It is not acceptable to constrain the function to work over only one network. So instead of making calls directly to the network as would be done in System V Release 2 or 4.2 BSD. the function is implemented using STREAMS technology.

When the function requires a network service, it sends a service request downstream. These service requests can be fielded by any module that understands them. So, the distributed process manager can work in an OSI network or a DARPA network or an SNA network. This same concept can be used at any layer in the reference model. For example, a customer can migrate from MAP protocols to Technical Office Protocols (TOP) simply by substituting an 802.3 CSMA/CD protocol module for the functionly equivalent 802.4 Token Bus module.

## Protocol Portability

One of the nice things about STREAMS is that it isolates the protocol module from its surrounding environment. This leads to protocol portability. A protocol module can be ported to another machine by a STREAMS environment on that machine and providing modules downstream that provide the service that the module requires. In Figure 3, and X.25 module has been ported to Machine B by providing a STREAMS environment on B and writing a STREAMS device driver on Mahcine B that controls the LAPB hardware that X.25 needs. Since the device driver is hardware dependent it is not portable.

This protocol portability has major ramifications for computer manufacturers. It means that all protocols, except those implemented in hardware, are directly portable across all machines that use the UNIX system.

This also has major significances for the software industry it means that ISVs can produce protocol modules that are portable to any vendor's machine in the same way that System V applications are currently portable.

## Internetworking

Internetworking illustrates the power and flexibility of STREAMS. Internetworking is the problem of connecting several dissimilar physical subnetworks into one logical network. The solution is to use a common internetworking protocol to route data from one subnetwork to another until it reaches its final destination.

There are two popular styles of internetworking. In the connectionless school, the internetworking protocol routes individual pieces of data from one subnetwork to another. All internetworking protocol asks is that the subnetwork be able to send the datagram from one edge of the subnetwork to the other (that is, for example, to send a datagram from one node on an Ethernet to another).

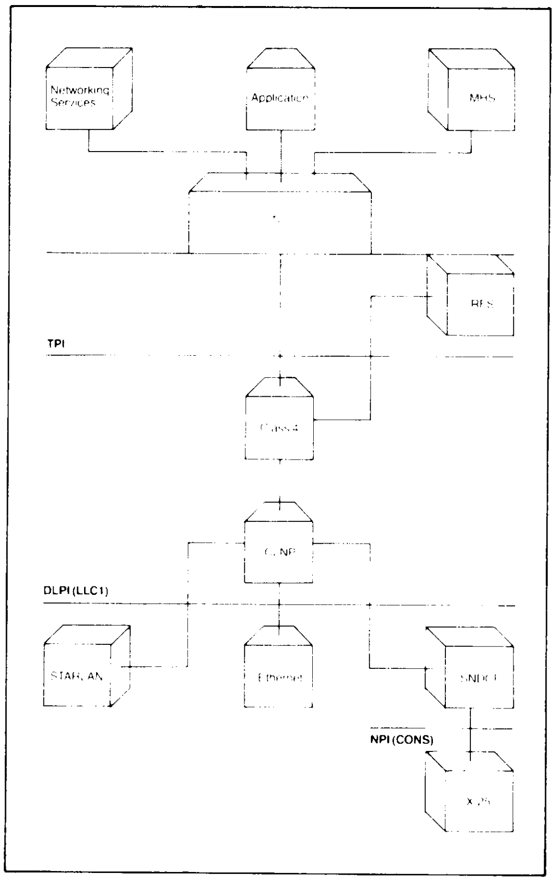


**Figure 4.** OSI Today
An implementation of OSI Transport X Network Layer Protocols in STREAMS provides the ability to internetwork between STARLAN and ETHERNET style local area networks and X.25 wide area networks. CLNP is the connectionless internetworking protocol, and ISO Class 4 transport protocol is the transport provider. Standard UNIX system applications as well as RFS and other network services access the transport provider directly through the TLI.

In the connection-oriented school of internetworking, the internetworking protocol expects the subnetworks to provide reliable virtual circuits. The job of the internetworking protocol is simply to weld the individual virtual circuits to an end-to-end path.

The bottom portion of Figure 4 shows a connectionless internetworking scheme as used in TCP/IP, XNS and MAP/TOP. The way that

STREAMS technology is used would be the same in a connection-oriented example. In this connectionless scheme, CLNP (Connectionless Network Protocol) is the internetworking protocol. It provides a Connectionless Network Services (CLNS)—that is an unreliable end to end datagram service. It relies on each of its subnetworks to provide a single-hop datagram service as specified in the IEEE 802.2 local area network standards.[8] CLNP makes routing decisions based on the internet address of the target end system. It decides, based on this address, where to send the datagram to take it to the next stage towards its destination. For example, if the datagram originated on the 802.3 (CSMA/CD) LAN and needed to go across the X.25 WAN before reaching the target on the 802.4 (Token Bus) LAMN, CLNP on the router on the 802.3 LAN would send the datagram across the X.25 subnetwork to the router on the 802.4 subnetwork. The implementation of this routing function is made easy using STREAMS.

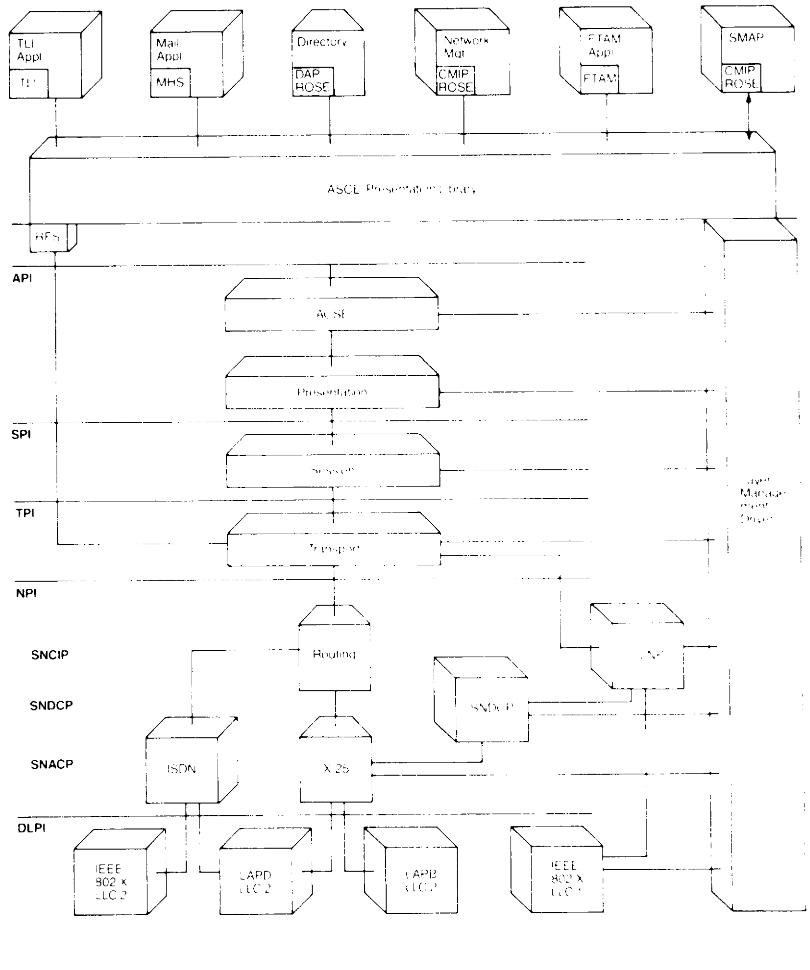


**Figure 5.** OSI Long Term
STREAMS will enable ISDN to be either integrated as a subnetwork under CLNP or be used as a provider of network layer service. The upper layer Protocols will be implemented directly over the Transport Provider Interface.

CLNP maintains a table that maps an internet address into the stream leading to the appropriate subnetwork. When it receives a datagram it looks at the internet address and sends the datagram down the appropriate stream with an 802.2 single hop datagram service request. It doesn't care that these subnetworkings are physically different. All it knows is that each one supports the 802.2 service interface. The drivers for the LANs provide this service directly; while for X.25 a function must be provided that maps the single hop datagram service requests into the virtual circuit requests that X.25 understands. This convergence function (SNDCF in Figure 4) is a virtual circuit manager. When it receives a datagram service request, it looks to see if it has a circuit open to the target system. If it does, it sends the datagram directly over the circuit. If it doesn't, it must first open a virtual circuit. For the sake of economy, it may wish to shut down some less frequently used circuit. This convergence function is specified by the OSI reference model. The good thing about STREAMS is that neither CLNP nor X.25 is affected by this convergence. CLNP treats the combination of X.25 and the SNSDF as a connectionless subnetwork. X.25 keeps a pure connection-oriented view of the world. The same X.25 module could be used to participate in a connection-oriented ISDN or SNA internetworking scheme or be used directly to provide reliable virtual circuits over a public Packet Switching Network (PPSN).

## Protocol Architectures

The System V AOE specifies support of the three major commercial protocol suites: OSI, TCP/IP and SNA. The following sections described how these protocol suites are implemented in STREAMS.

### OSI

Figure 4 shows a STREAMS implementation of the currently stable OSI protocols. It provides the ability to internetwork between STARLAN and ETHERNET style LANs and X.25 PPSN WANs.

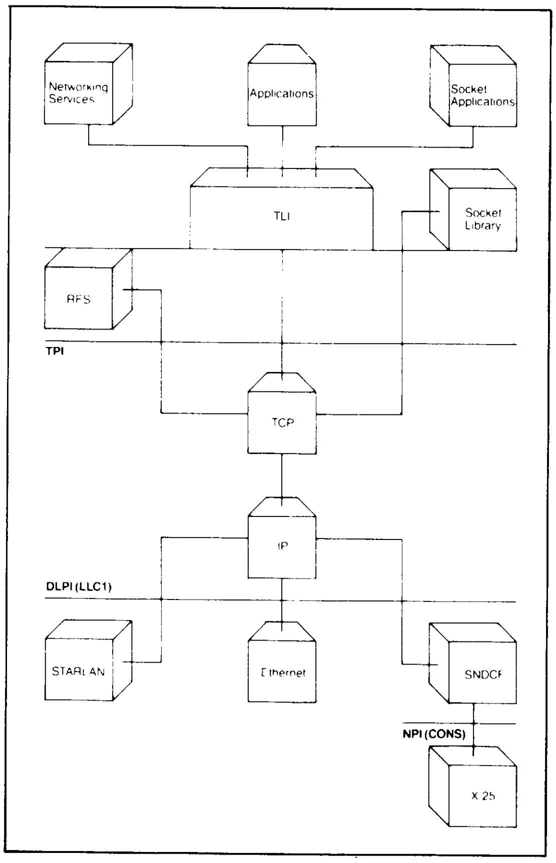


**Figure 6.** TOP/IP Protocols
The Transmission Control Protocol/Internet Protocol (TCP/IP) also provide internetworking between local area networks like STARLAN and ETHERNET and wide area networks like Telnet and ftp are implemented directly to the Transport Library Interface. This configuration also supports the Sockets applications in 4.2 BSD through a Sockets library interface.

Each subnetwork is treated as an 802.2 link layer provider (with an SNDCDF over the X.25 provider). CLNP is the internetworking section. The ISO Class 4 Transport protocol is the transport provider. Standard UNIX system applications, as well as RFS, uucp and other networking services access Class 4 directly through TLI. Upper layer applications such as the MHS electronic mail application are implemented at the UNIX system user level on top of TLI. This is because these protocols are still being refined in the standards bodies and are subject to frequent changes.

In the next few years there will be major enhancements to commercial OSI implementations. First, ISDN will be incorporated as an important class of network service. Second, the upper layer protocols will stabilise and migrate to the System V kernel. Figure 5 shows how these changes can be accommodated gracefully through the flexibility of STREAMS. ISDN can either be integrated as a subnetwork under CLNP or be used directly as a reliable and efficient provider of Network Layer service. The upper layer protocols will be implemented directly over TPI without breaking compatability with existing applications or networking services.

## TCP/IP

The implementation of the DARPA protocols in Figure 6 looks very much like the near term OSI implementation in Figure 4. TCP is the transport provider and IP the internetworking protocol. Note that the subnetworks remain precisely the same and it is possible to run OSI and TCP/IP over them simultaneously. TCP/IP has no upper layer architecture, so the networking services such as telnet or ftp are implemented directly to TLI. UNIX system applications and services such as uucp and RFS talk to TCP through TLI. In this configuration the sockets library for compatibility with BSD 4.2 is also supported.

## SNA

UNIX System vendors must be able to communicate effectively with SNA. There are three things that a UNIX system machine should be able to do with an SNA network. First, it must be able to communicate with another UNIX machine across and SNA network and support standard UNIX system applications and networking services. In addition, the UNIX system must be able to access information that resides on the IBM architecture mainframes. To do this, it must be able to speak the higher-level SNA protocols such as LU 6.2. Finally, to enable mainframe applications to migrate to UNIX system, 3270 style synchronous terminal anywhere in the SNA network must be able to log on the UNIX system and access applications in the accustomed manner.

The long term solution to the problem is to implement SNA Host (PU Type 5) host services on the UNIX system. Figure 7 shows a potential implementation of PU Type 5 SNA node under STREAMS. The heart of the architecture is a STREAMS-based implementation of the Path Control protocols. These protocols work either over SDLC lines or over X.25 wide area networks.

Each of the three modes of communication are implemented by accessing this STREAMS based Path Control transport provider. The System V applications and services access the LU6.2 basic conversation verb Transport Provider directly through TLI just as they did with TCP/IP. LU 6.2 is implemented as a STREAMS protocol module directly over Path Control. Applications and higher level services such as DIA can be implemented over the Common Transaction Processing interface library. The 3270 terminal traffic is handled by a protocol module that knows how to crack and format 3270 screen images. Applications can paint 3270 screens through a library interface similar to the current curses interface to asynchronous terminals.

## References

1.D.M. Ritchie, A Stream Input-Output System, AT&T Bell Laboratories Technical Journal, Vol. 63, Issue 8, October 1984.

2.Olander, David J. McGrath, Gilbert J. & Israel, Robert K.,. I A Framework for Networking in System V, Proceedings from 1986 Summer Usenix Conference, Usenix Association, June 1986.

3.UNIX System V Release 3 STREAMS Programmer's Guide, AT&T , June 1986.

4.CCITT Recommendation X.200,.I Reference Model of Open System Interconnection for CCIT Applications, 1984.

5.ISO IS 8072, Information Processing Systems - Open Systems Interconnection - Transport Service Definition, 1984.

6.System V Interface Definition, AT&T, 1986

7.Rifkin, A.P., RFS Architectural Overview, Proceedings of the 1986 Summer Usenix Conference, Usenix Association, June 1986.

8.ANSI/IEEE Standard 802.2, Logical Link Control, IEEE/Wiley-Interscience, 1984.

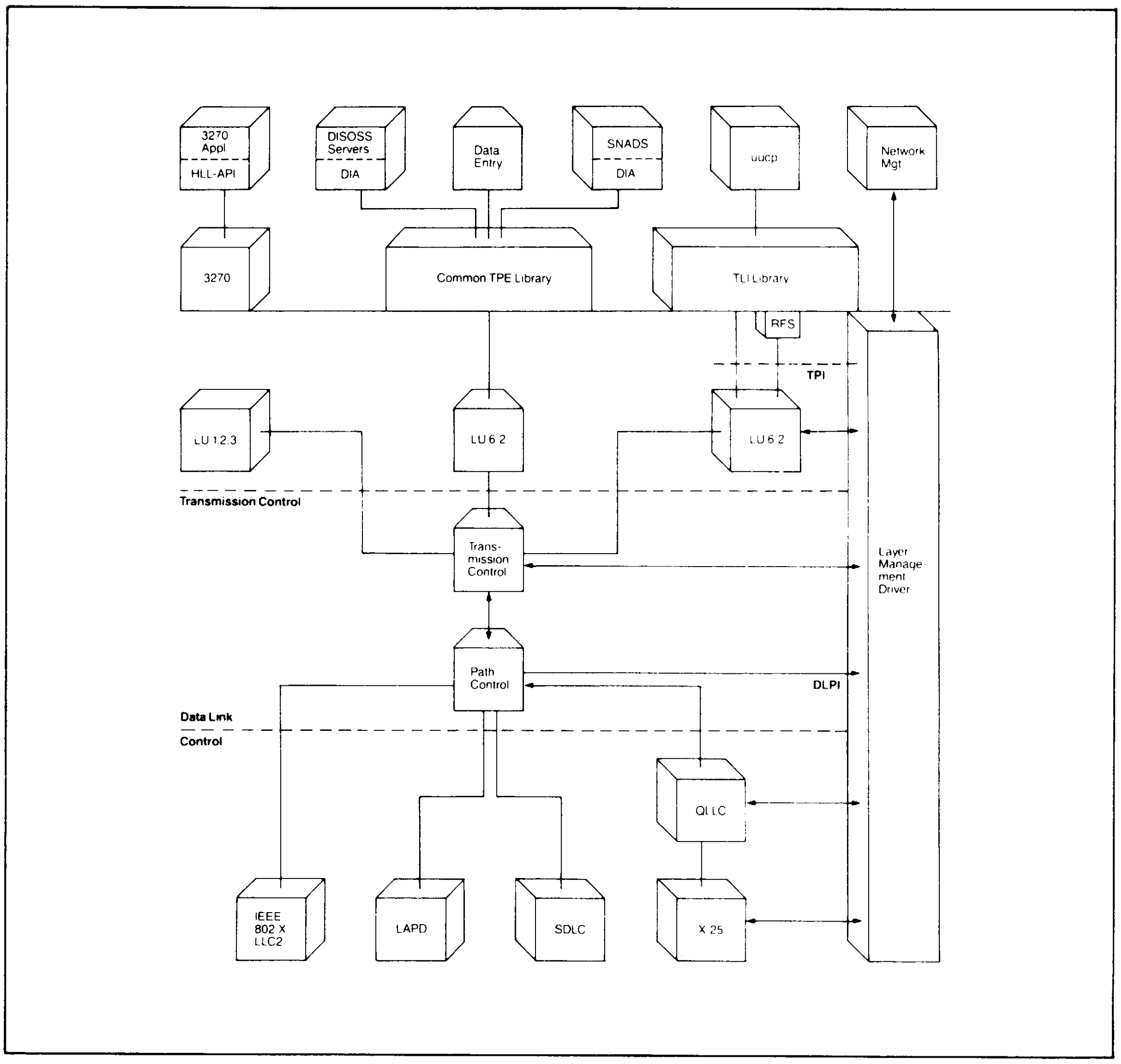


**Figure 7.** SNA Long Term
In the long term, the UNIX system must be able to communicate with IBM's System Networking Architecture (SNA). This will be accomplished by a STREAMS-based implementation of the Pat Control protocols. The UNIX System V applications and services access and LU.62 transport provider directly through TLL.

# UNIX Security Workshop

Organised by the UK UNIX systems User Group

This one-day workshop was held in February 1989 at the Institute of Education, University of London. The proceedings are now available on a three hour VHS video. Neil Todd (GID Ltd) was in the chair and the speakers are:

— o —

Russell Brand (Lawrence Livermore National Labs)
*HACKMAN: A Systematic Study of Real Computer Security Holes*

— o —

Piete Brooks (Cambridge University)
*Experiences at Cambridge*

— o —

Lindsay Marshall (Newcastle University)
*666: The Mark of the Beast*

— o —

Chris Milsom (UNIX Europe Ltd)
*System V/MLS*

— o —

Jim Reid (Strathclyde University)
*An Analysis of the Internet Worm*

— o —

William Roberts (Queen Mary College)
*The Sun Yellow Pages System*

— o —

Andy Rutter (The Instruction Set)
*Secure RPC*

— o —

Mario Wolczko (Manchester University)
*Some Myths and Facts about UNIX Security*

— o —

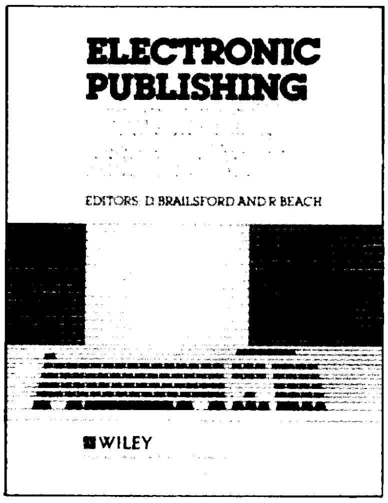
The cost is £50.00, including post & packing. The price does not include VAT, which is payable if you order from the UK. Copies of the tape can be ordered from (you will be sent an invoice):

Birkbeck College Video Services
Malet Street
London WC1E 7HX
England
(+44) 1 631 6351

EDITORS

**David F. Brailsford, BSc, Phd, FBCS**
Department of Computer Science,
University of Nottingham,
University Park,
Nottingham NG7 2RD, UK
Tel: (0602) 506101
Fax: (0602) 588138

**Richard J. Beach, Phd**
Computer Science Laboratory,
Xerox PARC, 3333 Coyote Hill Road,
Palo Alto, CA 94304, USA
Tel: (415) 494 4822
Fax: (415) 494 4022


ELECTRONIC PUBLISHING

EDITORS D BRAILSFORD AND R BEACH

WILEY

SUBSCRIPTION DETAILS
Volume 2 (1989) 4 issues
UK £80.00 Elsewhere US$144.00
Subscription prices include postage/air-speeded delivery

A special price of UK £60.00 or Elsewhere US$108.00 is available for personal subscriptions mailed to a home address and paid for by personal cheque or credit card.

If after receiving the first issue you are not satisfied with EP-odd, you may cancel your subscription and receive a full refund; the first issue will remain yours to keep.

*Nb. This guarantee applies until July 1st. Your written cancellation must be with us by July 21st.

WILEY

JOHN WILEY & SONS LTD

## SUBSCRIBE NOW

## No Risk... Money back Guarantee

# ELECTRONIC PUBLISHING

# ORIGINATION DISSEMINATION AND DESIGN

## AIMS AND SCOPE

- Structured Editors
- Authoring Tools
- Hypermedia
- Document Bases
- Document Display
- Electronic Documents
- Integrating of Text and Illustrations
- Typeface Design
- Imaging

EP-odd encourages articles which help to integrate techniques from all areas in electronic publishing

---

Please send me a FREE Sample copy of
**Electronic Publishing: Origination, Dissemination and Design**

Name____________________________________

Address__________________________________

________________________________________

________________________________________

________________________________________

Please return to Donna McMahon,
John Wiley & Sons Ltd, Baffins Lane, Chichester, West Sussex PO19 1UD, UK

## Window Systems

*William Roberts*
*liam@qmc-cs.uucp*

*Department of Computer Science*
*Queen Mary College*
*London, UK*

This is the second article in the series on Window Systems.

This article has been written by *Michael Gehret (mg@xsoft.uucp)* of *X/software, Grönenbach, West Gernamy*

# X/ST/window

*Michael Gehret*
*(mg@xsoft.uucp)*

After studying computer science and economics and graduating Michael Gehret founded X/software. The 'X' and the slash stand for UNIX, of course: the company has specialised in the development and distribution of system software for UNIX computers and for computers connected to UNIX machines. Meanwhile, 'X' has a second meaning: "a window system called X, not a system called X-Windows". A major interest lies with the X Window System—from the X Window server through libraries and toolkits upto adaptation of software to X Window.

## Introduction

UNIX and windows? From today's viewpoint, the name 'X' seems to be more programmatic than expected by its founders (Robert W. Scheifler and Jim Gettys state the descent from the 'V' and 'W' systems [1]). The X Window System is the graphics system, which comes closest to the original UNIX philosophy (modularisation, tools, distribution). However, one has to notice the questionable tendency to over-featuring and complexity in the X world, too.

Nach dem Abschluß des Studiums der Wirtschaftsmathematik gründete Michael Gehret die Firma X/software. Das 'X' und der Schrägstrich stehen—natürlich—für UNIX. Das Unternehmen ist spezialisiert auf Entwicklung und Vertrieb von Systemsoftware für UNIX-Rechner und für mit UNIX-Systemen vernetzte Computer. Inzwischen hat 'X' eine zweite Bedeutung: "a window system called X, not a system called X-Windows". Ein Tätigkeitsschwerpunkt der X/software ist das X Window System—vom X Window Server über Büchereien und Toolkits bis zur Anpassung von Software an den neuen Standard.

## Einführung

UNIX und Fenster? Der Name 'X' scheint mir aus heutiger Sicht programmatischer als von den Erfindern gedacht (Robert W. Scheifler und Jim Gettys beschreiben die Abstammung von den Systemen 'V' und 'W' [1]). Das X Window System ist das Grafiksystem, das der ursprünglichen UNIX-Philosophie (Modularisierung, Tools, Verteilung) am nächsten kommt. Allerdings ist auch im X-Umfeld die sehr bedenkliche Tendenz zum Feature-ismus und zur Unübersichtlichkeit zu bemerken.

William Roberts' first article of this column [2] compared the X Window System to other graphics systems and mentioned the advantages: manufacturer independence, quality, network transparency, flexibility and the developer's great freedom—but also responsibility—in making design decisions.

First, I'll summarise the design principles of the X Window System published in [1]. Then I'll introduce **X/ST/window**, our port of the X Window server to the ATARI ST, a powerful microcomputer. Finally, I'll outline the design of the system and the development environment.

## The X Window System...

The X Window System is a quasi-standard for graphics on UNIX, DEC and other computer systems. It was developed at the Massachusetts Institute of Technology in cooperation with renowned computer companies. The nine conceptual requirements of the X Window system distinguish it from other graphics systems:

*The system must be able to be implemented on various graphics hardware.*
Presently, the machines that are suitable for the X Window server range from simple monochrome raster monitors to colour subsystems with graphic processors and multiple monitors.

*The system must allow device independent applications programming.*
Applications should not have to be modified to suit the graphics hardware. Another aspect is that every graphics function defined by the system should work on virtually every supported display without questioning the capabilities of the X Window server.

*The system must be network transparent.*
The X Window server, responsible for user input and graphics output, and the application itself need not run together on one machine.

William Roberts hat im ersten Artikel dieser Kolumne [2] das X Window System mit anderen aktuellen Grafiksystemen verglichen und dabei die Vorteile genannt: Herstellerunabhängigkeit, hohe Qualität, Netzwerktransparenz, Flexibilität und große Entscheidungsfreiheit—aber auch Verantwortung—für den Software-Entwickler.

Ich fasse hier zunächst die Design-Konzepte des X Window System aus [1] kurz zusammen. Anschließend stelle ich **X/ST/window**, vor, unsere Portierung des X Window Servers für den ATARI ST, einen leistungsfähigen Microcomputer. Schließlich beschreibe ich kurz den Entwurf des Systems sowie die Entwicklungsumgebung.

## Das X Window System...

Das X Window System ist ein Quasi-Standard für Grafik auf Rechnern unter dem UNIX-Betriebssystem, DEC-Computer und andere Maschinen. Es wurde am Massachusetts Institute of Technology unter Beteiligung namhafter Rechnerhersteller entwickelt. Die konsequente und erfolgreiche Realisierung von neun Anforderungen an das Konzept unterscheidet das X Window System von anderen Grafiksystemen:

*Das System soll auf unterschiedlicher Grafik-Hardware implementiert werden können.*
Heute reichen die Geräte, für die X Window Server angeboten werden, von einfachen monochromen Rasterbildschirmen bis zu farbigen Subsystemen mit Grafik-Prozessoren und mehreren Bildschirmen.

*Das System soll die geräteunabhängige Programmierung von Anwendungen ermöglichen.*
Einerseits sollen Anwendungen unabhängig von der speziellen Grafik-Hardware sein. Andererseits sollen alle Grafikfunktionen des Systems von jeder Grafik-Hardware unterstützt werden, und zwar ohne besondere Eigenschaften des X Window Servers erfragen zu müssen.

*Das System soll netzwerktransparent sein.*
Der X Window Server, der für Benutzereingaben und Grafikausgaben zuständig ist, und die Anwendung müssen nicht notwendigerweise auf ein und derselben Maschine ausgeführt werden.

This means you can control a number crunching simulation program on a distant super computer from your X Window terminal and be able to watch the graphics results simultaneously. The applications and the X Window server communicate mainly over standardised network protocols—regardless of machine architecture or operating system.

*The system must support multiple applications running simultaneously.*
Due to network transparency, applications running on several computers within the network can display output on a single X Window terminal all at the same time.

*The system must be capable of supporting every kind of user interface.*
The X Window System doesn't specify one kind of user interface, it offers an implementation basis for various interfaces, depending on the type of application or the kind of user group.

*The system must support overlapping windows.*
Windows separate the different kinds of output from one or several applications, but also steer—in coordination with the mouse pointer—input to the chosen application. Overlapping windows use the monitor surface to its full advantage. The X Window System allows output to partially or totally obscured windows, which in turn allows applications to be absolutely independent of the present position and visibility of its windows.

*Every application must be able to use a hierarchy of resizeable windows.*
If an application needs to subdivide windows, it can make use of the X Window System's window abstraction for its own sub-windows. Every graphic object can own a window associating arbitrary many, including application specific, attributes.

So können Sie von Ihrem X Window Terminal aus ein rechenintensives Simulationsprogramm auf einem entfernten Supercomputer steuern und dessen Grafikausgabe betrachten. Die Anwendungen kommunizieren mit dem X Window Server meist über standardisierte Netzwerkprotokolle — unabhängig von Maschinenarchitekturen und Betriebssystemen.

*Mehrere Anwendungen sollen gleichzeitig bedient werden können.*
Aus der Netzwerktransparenz folgt, daß mehrere, auf verschiedene Rechner eines Netzwerkes verteilte Anwendungen simultan auf ein X Window Terminal ausgeben können.

*Das System soll offen sein für jede Form von Benutzerschnittstelle.*
Das X Window System legt nicht eine bestimmte Art von Benutzerschnittstelle fest, sondern bietet die Grundlage zur Implementierung der unterschiedlichsten Schnittstellen, je nach Anwendungstyp oder Benutzergruppe.

*Das System muß sich gegenseitig überlappende Fenster unterstützen.*
Fenster trennen einerseits die verschiedenen Arten von Ausgabe einer oder mehrerer Anwendungen und lenken andererseits—in Kombination mit dem Mauszeiger—die Eingabe zur Anwendung. Sich überlappende Fenster nutzen die Bildschirmfläche optimal aus. Das X Window System ermöglicht die Ausgabe in teilweise oder total verdeckte Fenster, so daß ein Anwendungsprogramm vollkommen unabhängig von der momentanen Lage und Sichtbarkeit seiner Fenster ist.

*Jede Anwendung soll eine Hierarchie von in ihrer Form veränderbaren Fenstern verwalten können.*
Benötigt eine Anwendung eine weitere Unterteilung ihrer Fenster, kann sie die Fensterabstraktion des X Window System s für eigene (Sub-)Fenster nutzen. Jedem grafischen Objekt kann ein Fenster mit beliebigen, auch anwendungsspezifischen, Attributen zugeordnet werden.

*The system must provide graphic functionality on a low abstraction level.*
The X Window System doesn't determine the kind of graphic interface used by the application. The simple functionality of the X Window server's graphics core system (2-D raster graphic, text) allows a higher abstraction level on the application side, such as object orientated interfaces (e.g., Athena Widgets, Andrew Toolkit, Xray) or graphic terminal emulation (e.g., Tektronix).

*The system should be extensible.*
An interface for the extension of the X Window server itself (e.g., PEX PHIGS 3-D graphics) is defined.

## ...for the

Presently, **X/ST/window** is the only port of the X Window server (version 11, release 3) on the ATARI ST.

Many ATARI STs are used as terminals on UNIX and DEC computers. The non-flickering screen, the DEC compatible keyboard, the possibility of running local applications and naturally the "Power Without the Price" are the advantages for this solution. However, the use of the ATARI ST is generally restricted to simple ASCII-terminal emulation and file transfer.

The ATARI ST's hardware is well suited for the implementation of the X Window server:

- The high performance MC68000 processor addresses a linear, unsegmented memory area of up to 4 Megabytes.
- The screen memory is accessed via a second memory bus, and its address is selectable.
- The graphic coprocessor Blitter relieves the main processor.
- Another processor deals with mouse and keyboard input.
- Extra hardware (e.g., a bigger monitor) can be attached via the processor bus or the DMA port.

*Das System soll Grafikfunktionalität auf einem niedrigen Abstraktionsniveau bieten.*
Das X Window System legt die Art der Grafikschnittstelle einer Anwendung nicht fest. Auf der einfachen Funktionalität des Grafikkerns des X Window Servers (2-D-Rastergrafik, Text) können auf der Anwendungsseite höhere Abstraktionsebenen wie objektorientierte Schnittstellen (z.B. Athena Widgets, Andrew Toolkit, Xray) oder Grafikterminal-Emulationen (z.B. Tektronix) implementiert werden.

*Das System soll erweiterbar sein.*
Für die Erweiterung des X Window Servers selbst (z.B. PEX PHIGS 3-D-Grafik) ist eine Schnittstelle definiert.

## ...für den

**X/ST/window** ist z.Z. die einzige Portierung des X Window Servers (Version 11, Release 3) auf den ATARI ST.

Sehr viele ATARI ST werden als Terminals an UNIX- oder DEC-Rechnern eingesetzt. Der vollkommen flimmerfreie Bildschirm, die DEC-kompatible Tastatur, die Möglichkeit, lokale Anwendungen zu fahren und natürlich der günstige Preis sprechen für diese Lösung. Allerdings bleibt die Verwendung des ATARI ST in der Regel auf einfache, zeichenorientierte Terminalemulation und Dateitransfer beschränkt.

Die Hardware des ATARI ST eignet sich sehr gut zur Implementierung eines X Window Servers:

- Der leistungsstarke Prozessor MC68000 greift auf einen linearen, unsegmentierten Adreßraum von bis zu 4 MByte zu.
- Der beliebig verschiebbare Bildschirmspeicher wird über einen zweiten Speicherbus adressiert.
- Der Grafik-Coprozessor Blitter entlastet den Hauptprozessor.
- Ein weiterer Prozessor verarbeitet Maus- und Tastatureingabe.
- Am Prozessorbus oder an der DMA-Schnittstelle kann zusätzliche Hardware, z.B. ein größerer Bildschirm, angeschlossen werden.

## Operating system, ...

The X Window server has very high demands on the computer's system software, which is generally UNIX. The ATARI ST's operating system, TOS, is does not have the necessary features for a direct implementation of the server. This is the reason why **X/ST/window** is based on **X/ST/multi**, a multi-tasking operating system that is totally compatible to TOS. The following concepts are used especially by **X/ST/window**:

- The process manager administers a number of processes with different priorities to be run quasi-parallel. Besides the X Window server process there are at least one protocol and two daemon processes running on the ATARI ST (see below).
- Processes are synchronised with semaphores.
- The inter process message system uses semaphores without polling and is functionally compatible to UNIX System V. Message queues are used by **X/ST/window** to enable the server to run independent of the time critical protocol process.
- The very efficient dynamic memory management supports the message system: Processes exchange the addresses of data areas, instead of the data areas themselves (shared memory and memory transfer). Data flow appears as a circulation of data areas from the network to the server and vice versa.

## ...graphics...

The partly newly developed graphics core system of **X/ST/window** supports monochrome raster graphics. Only a section of a virtual screen is to be seen, the dimensions can be chosen freely, only limited by the available main memory. The section is moved automatically the instant the mouse crosses the screen's borders.

Some of the graphic functions make use of the graphics coprocessor (Blitter), but don't neccessarily need it.

## Betriebssystem, ...

Der X Window Server stellt sehr hohe Anforderungen an die Systemsoftware des zugrundeliegenden Rechners—in der Regel handelt es sich um UNIX. Das Betriebssystem des ATARI ST, TOS, eignet sich nicht zur direkten Implementierung des Servers. **X/ST/window** basiert daher auf **X/ST/multi**, einem Multitasking-Betriebssystem, das zu TOS vollkommen kompatibel ist. Die folgenden Konzepte werden insbesondere von **X/ST/window** genutzt:

- Der Prozeßmanager verwaltet beliebig viele Prozesse, die quasi parallel mit verschiedenen Prioritäten ausgeführt werden. Auf dem ATARI ST laufen neben dem eigentlichen X Window Server mindestens ein Protokoll- und zwei Dämon-Prozesse (s.u.).
- Prozesse können über Semaphore synchronisiert werden.
- Das der Interprozeßkommunikation dienende Message-System ist funktional zu UNIX System V kompatibel und mit Semaphoren ohne Polling realisiert. **X/ST/window** verwendet *message queues* zur Entkoppelung des Servers vom zeitkritischen Protokollprozeß.
- Eine sehr effiziente dynamische Speicherverwaltung unterstützt das Message-System: Prozesse tauschen nicht Datenflächen, sondern Adressen von Datenflächen aus (*shared memory* und *memory transfer*). So stellt sich der Datenfluß in **X/ST/window** als Kreislauf von Datenflächen vom Netzwerk zum Server und umgekehrt dar.

## ...Grafik...

Der zum Teil neu entwickelte Grafikkern von **X/ST/window** realisiert monochrome Rastergrafik. Auf dem Monitor ist nur ein Ausschnitt eines virtuellen Bildschirms zu sehen, dessen Abmessungen frei wählbar und lediglich durch den Arbeitsspeicher begrenzt sind. Dieses Fenster wird automatisch verschoben, sobald der Mauszeiger eine Begrenzung des Ausschnitts überschreitet.

Einige der Grafikfunktionen nutzen den Grafik-Coprozessor Blitter. Er wird aber nicht vorausgesetzt.

**X/ST/window** can load a keyboard definition from the local mass storage or from a UNIX-computer within the network to meet national or special requirements. You can activate two keyboard layouts at the same time.

## ...and network

**X/ST/window** can communicate with one or more X applications over a network connection, this being mainly TCP/IP on the UNIX side. With the choice of daemon and protocol processes, the following options are possible:

- One or more ATARI STs are connected in a star-shaped manner to a UNIX computer over serial terminal lines (19200 Baud). A very safe and efficient protocol was implemented for this solution. The connection to TCP/IP is through a daemon process running on the UNIX computer, which simulates one or more X Window servers for applications throughout the network.
- The ATARI STs are linked to the UNIX machine by Ethernet interfaces. The Ethernet hardware (without cpip) takes the part of the serial lines mentioned above.
- The ATARI ST is used as a node within an Ethernet network under TCP/IP. This solution does not require additional software on the UNIX side.

The choice is made at the start of the X Window server, which is totally independent of the kind of network connection available. This way **X/ST/window** need not be modified when upgrading to new, more expensive and higher-performance network technology.

Font files can be managed either locally on the ATARI ST or centralised or distributed over the computers in the network, in ASCII, binary or `compress` format.

**X/ST/window** kann eine Tastaturdefinition vom lokalen Massenspeicher oder von einem Rechner des Netzwerks laden, um einem nationalen oder einem speziellen Tastatur-Layout zu entsprechen. Es ist auch möglich, gleichzeitig zwei Tastaturbelegungen zu aktivieren.

## ...und Netzwerk

**X/ST/window** kommuniziert über eine Netzwerkverbindung mit einer oder mehreren X Window Anwendungen. Auf der UNIX-Seite handelt es sich dabei meist um TCP/IP. Durch die Wahl verschiedener Dämon- und Protokollprozesse sind folgende Optionen möglich:

- Ein oder mehrere ATARI ST sind über serielle Terminalleitungen (19200 Baud) sternförmig mit einem UNIX-Rechner verbunden. Hierzu wurde ein sehr sicheres und effizientes Protokoll implementiert. Der Anschluß an TCP/IP erfolgt über einen Dämon-Prozeß, der auf dem UNIX-Rechner für im Netzwerk verteilte X Window Anwendungen einen oder mehrere X Window Server simuliert.
- Die ATARI ST werden über Ethernet-Schnittstellen mit einem UNIX-Rechner verbunden. Die Aufgabe der o. a. seriellen Leitung übernimmt hier die Ethernet-Hardware (ohne TCP/IP).
- Der ATARI ST wird als Knoten in einem Ethernet-Netzwerk mit TCP/IP betrieben. Diese Lösung benötigt auf der UNIX-Seite keine zusätzliche Software.

Die Auswahl erfolgt beim Start des X Window Servers, der vollkommen unabhängig von der Art der Netzwerkverbindung ist. So ist die einfache Migration von kostengünstigen zu leistungsfähigen sowie die Integration neuer Netzwerktechnologien ohne Änderung von **X/ST/window** möglich.

Zeichensatzdateien können auf dem ATARI ST lokal sowie zentral oder verteilt auf Rechnern des Netzwerks verwaltet werden, in ASCII-, binärem oder `compress`-Format.

A fileserver daemon runs in background to the X Window server. It provides UNIX-lookalike commands like `cp`, `mv`, `ls`, `pwd`, `cd`, `mkdir`, and `rmdir`. These commands affect the local filesystem of the ATARI ST and the remote virtual filesystems of connected hosts. For example, the command `cp unix_file c:\tos_file &` copies a file from the UNIX host to the ATARI ST—parallel to the graphics output of the X Window server. The command `print` prints an arbitrary file on the local printer.

## The situation presently...

After being presented at the Hanover CeBIT fair in March, **X/ST/window** is now available. The software needed for connections to TCP/IP has been ported to several UNIX computers. By the time you read this text, TCP/IP for the ATARI ST should be available. The software requires 2 MByte of main memory (i.e., ATARI ST Mega 2 or Mega 4), but no hard disk.

## ...and the future

**X/ST/window** will always be up to date with the latest X Window standard. In keeping with continual improvement, we will attempt to integrate suggestions coming from the user side. Thus, the server will run with big screens providing higher resolutions.

## Design...

As mentioned above, the ATARI ST's operating system, TOS, is not comparable to UNIX. Therefore, there are two ways to bring the X Window server to the ATARI ST: reducing the server's functionality, or implementing UNIX-lookalike capabilities on the ATARI ST. Of course, our server should be fully compatible to the X Window Protocol definition [3]. So we decided to go the second, harder, way. The multitasking system **X/ST/multi** allows to port UNIX-based software to the ATARI ST with only few modifications, and even to combine it with TOS software.

Im Hintergrund zum X Window Server läuft ein Fileserver-Dämon, der es erlaubt, UNIX-ähnliche Kommandos wie `cp`, `mv`, `ls`, `pwd`, `cd`, `mkdir`, `rmdir` abzusetzen. Diese Befehle wirken auf das lokale Dateisystem des ATARI STs wie—virtuell—auf die Dateisysteme der über das Netzwerk angeschlossenen Hostrechner. So kann z. B. mit `cp unix_file c:\tos_file &` eine Datei von einem UNIX-Rechner zum ATARI ST kopiert werden—im Hintergrund zur Grafikausgabe des X Window Servers. Mit `print` können beliebige Dateien lokal ausgedruckt werden.

## Der aktuelle Stand...

Nach der Vorstellung auf der CeBIT in Hannover ist **X/ST/window** jetzt erhältlich. Die Software zum Anschluß an TCP/IP wurde auf verschiedene UNIX-Rechner portiert. Zum Zeitpunkt der Veröffentlichung dieses Textes ist TCP/IP für den ATARI ST wahrscheinlich verfügbar. Die Software setzt z.Z. noch einen Arbeitsspeicher von 2 MByte voraus (ATARI ST Mega 2 oder Mega 4), eine Festplatte ist nicht erforderlich.

## ...und die Zukunft

**X/ST/window** wird immer dem neuesten Standard von X Window entsprechen. Parallel zu dieser laufenden Aktualisierung werden wir Anregungen aus dem Benutzerkreis verwirklichen, so z.B. die Unterstützung großer Monitore mit höherer Auflösung.

## Entwurf...

Wie bereits bemerkt, ist das Betriebssystem des ATARI STs, TOS, nicht mit UNIX zu vergleichen. So gibt es grundsätzlich zwei Wege, den X Window Server auf dem ATARI ST zu implementieren: entweder die Einschränkung der Funktionalität des Servers oder die Schaffung UNIX-ähnlicher Voraussetzungen auf dem ATARI ST. Selbstverständlich soll unser Server der X-Protokoll-Definition [3] entsprechen, wir haben uns also für die zweite, schwierigere Möglichkeit entschieden. Das Multitasking-System **X/ST/multi** erlaubt es uns, UNIX-basierte Software mit relativ geringen Änderungen auf den ATARI ST zu portieren und mit TOS-Software zu kombinieren.

Of course, the server's graphics parts needed extensive enhancements and modifications. I mention the important ones only:

- keyboard driver with autorepeat and loadable keymaps;
- mouse driver simulating the third, middle button;
- simulation of a hardware cursor in software, for greater efficiency on a very low level;
- implementation of the virtual screen manager;
- integration of the graphics coprocessor;
- optimisations and adaptations to the architecture of the MC68000.

The networking part —protocols for serial line and raw Ethernet, daemon programs, and the fileserver—is totally new. This software is supported by **X/ST/multi**'s message system.

## ...and development

Soon it became clear, that even the 4-MByte ATARI ST Mega 4 is not the appropriate development platform for the port of the X Window server: the MIT distribution of the software is based on BSD UNIX, and we wanted to use the extensive development environment of UNIX.

Now the software is maintained on a Hewlett-Packard 9000/330 workstation. This computer uses the MC68020 processor, contrary to the ATARI ST which is built around the first processor of this family, the MC68000.

Fortunately, HP-UX's C development environment contains compiler and assembler for the MC68000 and MC68010 processors. The generated code emulates the floating point coprocessor MC68881. We created driver programs for the special compiler and linker, and a program, which translates executable program files from the UNIX workstation's `a.out` format into a format loadable by the ATARI ST. The symbol table is translated, too, preserving the significance of long symbol names.

Die Grafik-Software des Servers erforderte natürlich umfangreichere Ergänzungen und Modifikationen. Hier seien nur die wichtigsten genannt:

- Treiber für die Tastatur mit Wiederholfunktion und ladbarer Tastaturbelegung;
- Treiber für die Maus mit Simulation des dritten, mittleren Mausknopfes;
- Simulation eines Hardware-Cursors in Software, zur Effizienzsteigerung auf einer sehr maschinennahen Ebene;
- Implementierung des virtuellen Bildschirm-Managers;
- Ansteuerung des Grafik-Coprozesors;
- Optimierungen und Anpassungen an die Architektur des MC68000.

Die Netzwerk-Seite —Protokolle für serielle Terminalleitung und rohes Ethernet, Dämonprogramme und der Fileserver—wurden auf der Basis des Message-System von **X/ST/multi** neu entwickelt.

## ...und Entwicklung

Es war recht bald klar, daß selbst ein ATARI ST Mega 4 mit 4 MByte Arbeitsspeicher nicht die geeignete Maschine für die Portierung des X Window Servers darstellt. Zum einen basiert die Software des MIT auf BSD UNIX und zum anderen sollten die umfangreichen UNIX-Tools bei der Entwicklung Verwendung finden.

Die Software wird z.Z. auf einer Hewlett-Packard 9000/330 Workstation weiterentwickelt. Dieser Rechner arbeitet mit einem MC68020 Prozessor, während der ATARI ST mit dem ersten Prozessor dieser Familie, dem MC68000, ausgerüstet ist.

Die C-Entwicklungsumgebung unter HP-UX beinhaltet jedoch einen Compiler und einen Assembler für die MC68000 und MC68010 Prozessoren. Der Floating-Point-Coprozessor MC68881 wird dabei emuliert. Wir entwickelten Treiberprogramme für den speziellen Compiler und den Linker, sowie ein Programm, das ausführbare Programmdateien vom `a.out`-Format der UNIX-Workstation in ein Format umwandelt, das vom ATARI ST geladen werden kann. Dabei wird auch die Symboltabelle gewandelt, und zwar unter Beibehaltung der Signifikanz langer Symbolnamen.

Thus we can let a symbolic debugger drive **X/ST/window** on the ATARI ST.

The system calls of the C library are transformed to correspondent TOS-system calls by the multitasking system **X/ST/multi**.

HP-UX, Hewlett-Packard's variant of UNIX is a "System V with BSD extensions". Nevertheless, the compilation of the core software and of the less portable contributions was rather straightforward. For some time we ran Hewlett-Packard's X10-server and X10-clients beside ported X11-clients controlled by a X11-**X/ST/window** simultaneously.

So können wir **X/ST/window** auf dem ATARI ST unter einem symbolischen Debugger betreiben.

Die Systemaufrufe der C-Bücherei werden unter dem Multitasking-System **X/ST/multi** auf die entsprechenden TOS-Systemaufrufe abgebildet.

HP-UX, die UNIX-Variante von Hewlett-Packard, ist eher an System V als an BSD orientiert. Trotzdem gab es bei der [bersetzung der *core*-Software wie auch der weniger portablen *contributions* wenig Probleme. Zeitweilig betrieben wir gleichzeitig einerseits X10-Server und -Clients von Hewlett-Packard und andererseits von uns portierte X11-Clients in Verbindung mit einem X11-**X/ST/window**.

## Conclusions

Once more, the ATARI ST turned out to be an underestimated powerful computer. Using extensive development tools one can implement sophisticated systems on top of suitable system software.

**X/ST/window** offers inexpensive graphics functionality and may substitute a workstation in a network. Of course, its performance is less than that of a high-end workstation. But often the demands on a graphics terminal are lower.

**X/ST/window** can be used as a UNIX terminal for text, too: the possibility to control multiple shells—even on different machines—at the same time from one terminal is very valuable.

## Zusammenfassung

Der ATARI ST erwies sich wieder als allgemein unterschätzter, leistungsfähiger Computer. Auf entsprechender Systemsoftware und mit leistungsfähigen Entwicklungswerkzeugen lassen sich anspruchsvolle Systeme realisieren.

**X/ST/window** bietet einen preisgünstigen Grafikarbeitsplatz und kann so eine Workstation in einem Netzwerk ersetzen. Selbstverständlich ist die Performance geringer als die einer Hochleistungs-Workstation. Oft werden jedoch nur geringere Anforderungen an einen Grafikarbeitsplatz gestellt.

**X/ST/window** kann natürlich auch als UNIX-Terminal für Textanwendungen verwendet werden: die Möglichkeit, gleichzeitig mehrere Shells—sogar auf verschiedenen Rechnern—von einem Terminal aus zu kontrollieren, ist sehr wertvoll.

## Bibliography — Literatur

[1] Robert W. Scheifler, Jim Gettys: The X Window System, in: ACM Transactions on Graphics, Vol. 5, No. 2, April 1986

[2] William Roberts: Introduction to Window Systems, in: EUUG Newsletter, Vol. 9, No. 1, Spring 1989

[3] Robert W. Scheifler: X Window System Protocol, MIT

## C—The Effective Solution, An Intensive Video Course

Mick Farmer and Richard Murphey, Chartwell-Bratt, 1988, 2 Volumes plus 2 tapes (VHS—other formats also available), ISBN 0-86238-190-8 and 0-907904-13-0. (UK) Price £850, Soft Back, 186 pp, size: (text book) A5, (work book) A4.

Reviewed by Jane Spreull of the Agricultural and Food Research Council Computing Centre.

This is a video based self-paced course in 'C'. It assumes only basic programming knowledge in other languages, and covers all the standard 'C' commands and most of the more common library functions.

The package contains a video tape, a workbook (*C: The Effective Solution Workbook*), and a textbook (*The Intensive 'C' Course*). The video tape consists of 10 units, each lasting about 30 minutes.

The Textbook is well structures with many example code fragments illustrating the use of the language in each section, giving the reader the 'flavour' of the 'C' language. The language presented is a standard version with none of the usual extensions which abound particularly in the PC 'C', and this I would consider to be a gret advantage (portability, availability of compilers, and the like).

The video presentation was good with changes in the viewing angles, and separate fill screen displays for 'C' text adding interest (assuming there is such a thing as interest in a programming language). The use of the screen layout for the demonstrations was clear but at times not very obvious. The workbook clarified the points bt a videotape player with a remote controller with a 'Stop' button is essential to give time for the script to be consulted. It is clear that many viewings of the tape mixed with some practical work would be needed to get the most out of the tape—you are, after all, trying to learn a new language.

The tone of voice used by Mick Farmer was modulated in a pleasing manner giving the impression of a well practiced and clear presentation. This leads to a feeling that the information given is complete and without errors. The presentation was very good compared with other video courses where the information was presented in such a monotone that it became irritating after a very short period of time.

The Workbook is *not* a simple transcript of the video but combines with it to give a package that provides a very good basis on which to build real 'C' programming expertise.

The course is available now from:

Birkbeck College Video Services
Malet Street
London WC1E 7HX
England
(+44) 1 631 6351

it will soon be available from your favourite book shop.

## UNIX-Communications

Bart Anderson, Bryan Costales, and Harry Henderson, The Waite Group (A Division of Macmillan), 1987, Price $29.50, 542 pp.

Reviewed by Anke Goos of University of Dortmund, IRB <anke@unido.uucp>.

### Introduction

"This book is bad", said an experienced News-Postmaster, weighing about one kilogram of paper in his hand, "our postmaster-team could have written this as well". As I had the experience of waiting a long time for one article about the News systems from this very person, my estimation is: "Some knowledge in my hand is better than inside others' heads". Especially as the head of this person has already left us for "serious professional work". There is a lot of information in those 540 pages of 'UNIX Communications' which has been compiled by three authors of the Waite Group.

In the context of this book UNIX Communications means three major chapters about 'Mail', the news conferencing systems of 'Usenet' and 'UUCP'. There are more than 200 pages dedicated only to the last topic. The book is not written for people who like digging in UNIX manuals. Instead it's a book for newcomers and interested people, beginning at point zero. BTW: zero is an introduction to UNIX as an operating system in 20 pages.

## Mailing First...

At this stage the innocent UNIX-fan is initiated in the ABC of mailing step by step through well-structured units and a lot of illustrations. This how-to-mail is presented by the author Bert Anderson, following the example of mailx as the mailing software for System V machines and mail for Berkeley 4.2.

But to get to those 'elitist' :-) functions like including a file or a message you have to survive thirty pages to reach a chapter on 'Advanced Mailer'. I would recommend that you do not follow each chapter and paragraph, but pick out all the functions that you may need for a happy mailing life. This might already be the neat explanation for manipulation of your *.mailrc* file for a CC of an automatic copy of your mail to people other than the recipient.

What's missing in this first part about mailing? Well, for example, a comparison to other mail user agents like the screen-oriented PD program Elm. Or other advanced systems like MH, recommandable if you really have to handle a lot of e-mail. Furthermore, there's no preparation on the 'other' maybe shocking form of communication in daily e-mail life like:

>The indentation of those sentences you are referring to..<
Yes, that's what I'm thinking of ...
>The neat smileys :-), turn your head at the left side by 90 degrees, thank you. You got it? No ?-) Too bad :-(.

The author shows a reluctance to deal with the complex topic of addressing in UNIX networks, or even through to other networks—just as if all that exists is the old-style bang-addressing of UUCP (*host1!host2!!user*) throughout networld—the book was written in 1987. The concentration on the more chaotic American UUCP network is also shown by a whole chapter on how to construct a path of hosts to reach your end-user. Sometimes the Europeans are a bit advanced :-).

### Usenet for the Advanced Mailer

You might have imagined that network communication is explained in the 'Usenet' chapter by Harry Henderson. It is, but it is focussed on the computer conferencing system of the 'readnews' and the topological network of the Usenet backbone, secondary feeders and end-user hosts. A lot of standard information out of the newsgroups is summarised. Extended paragraphs inform you about modern user interfaces for reading the 'news', like *rn* or the screen-oriented *vnews*. You might even find interesting details, like changing the the newsgroup list of a posting or the deleting of articles which have already been posted. I estimate some details are useful even for an advanced news reader.

*Last but not least: UUCP*

This comprehensive part covering UUCP migrates away from the needs of an end-user in the direction of the programmer, system administrator or curious UNIX-wizard. Although this information is already well prepared and explained, there are more hard facts about the UUCP-Program in general and in detail.

Examples are the copying of files by UUCP and UUSEND, UUID and UUPICK or remote execution on other hosts by uux, Status Reports and Job Control or CU for login on other machines. Or the copying and packing of files, directories and binaries with shar, tar or cpio (Copying Archives and Binary Files).

In the very end this book becomes a typical American 'How-to-do-Book'. In the way of 'We are sending a mail' the cooperation of all UUCP tools is explained step by step. This is not too bad as an overview. But we are not yet finished.

Some critics remain: Of course it would have been better to have two or three different handbooks about the three different topics. At least to decrease the personal barrier. Although you might fall asleep, the book is not really suitable for bed time reading. Sometimes it goes into too much detail. Other criticisms? Yes. There is no word about EUnet in the whole book. Neither in the index nor in the description of the backbone topology. Europe is simply considered to be a part of Usenet and referenced by some obsolete backbone names. But you can't have everything. Unless you want to write a book for yourself. Any volunteers? EUnet is still searching for good documentation to be included in a starter-kit for new members...

# Call For Papers

## AUUG '89

**Australian Unix systems User Group**

**Conference and Exhibition 1989**

**August 8–11 1989, Sydney, Australia**

### Summary

The 1989 Conference and Exhibition of the Australian UNIX systems User Group will be held on Tuesday 8th – Friday 11th August 1989 at the Hilton Hotel in Sydney, Australia. Tutorial sessions will be held on Tuesday 8th, and the conference proper from Wednesday the 9th to Friday the 11th. The conference theme is

*No one ever got fired for buying UNIX.*

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

| | |
|---|---|
| **Dennis Ritchie** | Bell Laboratories |
| **James Gosling** | Sun Microsystems |
| **Sunil Das** | City University, London |
| **Ross Bott** | Pyramid Technology Corporation |

### Registrations

Exhibition and Conference Registrations are being handled by Australian Convention Management Services, and you should contact them with any enquiries about exhibition space, registrations, accommodation, and so forth:

| | | | |
|---|---|---|---|
| ACMS | Phone: | International | +61 2 3324066 |
| P.O. Box 468 | | | |
| Paddington NSW 2021 | | | |
| Australia | | | |

### Papers

Papers on topics related to commercial, government, and non-technical uses of UNIX (and also on non–commercial and technical aspects of the UNIX system) are now invited.

| | | | |
|---|---|---|---|
| Peter Barnes | Phone: | International | +61 7 3772907 |
| AUUG 89 | | National | 07 3772907 |
| Computer Science | Fax: | International | +61 7 3710783 |
| University of Queensland | | National | 07 3710783 |
| St. Lucia | Telex: | UNIVQLD AA40315 | |
| Queensland 4067 | ACSnet: | pdb@uqcspe.cs.uq.oz | |
| Australia | UUCP: | uunet!munnari!uqcspe.cs.uq.oz!pdb | |
| | ARPA: | pdb%uqcspe.cs.uq.oz@uunet.uu.net | |

Timetable

| | |
|---|---|
| Conference tutorials | 8th August |
| Conference and Exhibition | 9th–11th August |

# Calendar of UNIX Events

This is a combined calendar of planned conferences, workshops, or standards meetings related to the UNIX operating system.

If you have a UNIX related event that you wish to publicise then contact either John Quarterman at *jsq@longway.tic.com* or Alain Williams at *addw@phcomp.co.uk* giving brief details as you see below.

Abbreviations:

| | |
|---|---|
| C | Conference |
| G, MD | Gaithersburg, Maryland |
| S | Symposium |
| T | Tradeshow |
| U | UNIX |
| W | Workshop |
| UG | User Group |

| year mon days | conference | (sponsor,) (hotel,) location |
|---|---|---|
| 1989 May 1-2 | USENIX Transactions | W Pittsburgh, USA |
| 1989 Jun | NZSUGI | New Zealand |
| 1989 Jun 7-9 | Italian U C | i2u, Milan, Italy |
| 1989 Jun 7-9 | COMUNIX | /usr/group/UK, |
| 1989 Jun 7-9 | Eur. U User Show | EMAP Int. Exh., Alexandra Palace, London |
| 1989 Jun 12-16 | USENIX | Hyatt Regency, Baltimore, MD |
| 1989 Jun 19-23 | ICX89 | USM, IEEE, Valparaiso, Chile |
| 1989 Jun 27-28 | UKUUG | Glasgow |
| 1989 Jul 10-12 | 13th JUS UNIX Symposium | Tokyo, Japan |
| 1989 Jul 10-14 | IEEE 1003 | San Francisco, CA |
| 1989 Jul 26-28 | IETF | IAB, Stanford, Stanford CA |
| 1989 Aug 8-11 | AUUG | Hilton Hotel, Sydney |
| 1989 Sep 7-8 | Sun UK UG | C Manchester, UK |
| 1989 Sep 7-9 | USENIX Sys Admin | W Austin, TX, USA |
| 1989 Sep 18-22 | EUUG | WU Vien, Vienna, Austria |
| 1989 Sep 19-22 | ACM SIGCOMM | Austin, TX |
| 1989 Sep 25-29 | GUUG | CT Wiesbaden, Germany |
| 1989 Oct 16-20 | IEEE 1003 | Brussels, Belgium |
| 1989 Oct | UNIX Expo | New York, NY |
| 1989 Oct 5-6 | USENIX -Distrib Proc | W Florida, USA |
| 1989 Oct 31-Nov | 2 IETF | IAB, U. Hawaii, Honolulu, HI |
| 1989 Nov 1-3 | UNIX EXPO | Javits Conv. C, New York, NY |
| 1989 Nov 6-10 | DECUS S | Anaheim, California |
| 1989 Nov 9 | NLUUG C | The Netherlands |
| 1989 Nov 9-10 | 14th JUS UNIX Symposium | Osaka, Japan |
| 1989 Nov 16-17 | USENIX Graphics | W California, USA |
| 1989 Nov 24 | AFUU C | Paris, France |
| 1989 Dec 5-6 | JUS UNIX Fair 89 | Tokyo, Japan |
| 1990 Jan | U in Gov. C&T | Ottawa, ON |

| | | |
|---|---|---|
| 1990 Jan 22-26 | USENIX | Washington, DC |
| 1990 Jan 23-26 | UniForum | Washington Hilton, Washington, DC |
| 1990 Jan 29 | IEEE 1003 | New Orleans, LA |
| 1990 Feb 6-8 | IETF | IAB, (FSU, Talahassee, FL) |
| 1990 Mar 27-30 | AFUU | Paris, France |
| 1990 Apr | IEEE 1003 | Montreal, Quebec |
| 1990 Apr 23-27 | EUUG | Munich, Germany (tentative) |
| 1990 May 2-4 | IETF | IAB, (U. Washington, Seattle, WA) |
| 1990 May 7-11 | DECUS S | New Orleans, Louisiana |
| 1990 May | U 8x/etc C&T | /usr/group/cdn, Toronto, ON |
| 1990 Jun 11-15 | USENIX | Marriott, Anaheim, CA |
| 1990 Jul 31-Aug | 2 IETF | IAB, ?, not in North America |
| 1990 Autumn | EUUG | south of France |
| | | |
| 1991 Jan 21-25 | USENIX | Dallas, TX |
| 1991 Jan 22-25 | UniForum | Infomart, Dallas, TX |
| 1991 Feb | U in Gov. C&T | Ottawa, ON |
| 1991 Spring | EUUG | Tromso?, Norway (tentative) |
| 1991 May | U 8x/etc C&T | Toronto, ON |
| 1991 Jun 10-14 | USENIX | Opryland, Nashville, TN |
| | | |
| 1992 Jan 20-24 | USENIX | Hilton Square, San Francisco, CA |
| 1992 Jan 21-24 | UniForum | Moscone Center, San Francisco, CA |
| 1992 Jun 8-12 | USENIX | Marriott, San Antonio, TX |
| | | |
| 1993 Jan | USENIX | northeast North America |
| 1993 Mar 2-4 | UniForum | Washington, D.C. |

## Organising Bodies

NIST/NBS/POSIX
Roger Martin
National Institute of Standards
and Technology
Technology Building, Room B266
Gaithersburg, MD 20899
+1-301-975-3295
+1-301-975-3295
rmartin@swe.icst.nbs.gov

IEEE Computer Society
P.O. Box 80452
Worldway Postal Center
Los Angeles, Ca. 90080

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

/usr/group/cdn
241 Gamma St.
Etobicoke, Ontario M8W 4G7
Canada
+1-416-259-8122

Tracy MacIntyre
Exhibition Manager
EMAP International Exhibitions Ltd.
Abbot's Court
34 Farringdon Lane
London EC1R 3AU
United Kingdom
+44-1-404-4844

AUUG
P.O. Box 366
Kensington
N.S.W. 2033
Australia
uunet!munnari!auug
auug@munnari.oz.au
+61 3 344 5225

AMIX, c/o IPA
P.O. Box 919
Ramat-Gan
Israel, 52109
+972-3-715770
+972-3-715772
amix@bimacs.bitnet
amix@bimacs.biu.ac.il

Japan UNIX Society (JUS)
#505 Towa-Hanzomon Corp. Bldg.
2-12 Hayabusa-cho
Chiyoda-ku, Tokyo 102
Japan
bod%jus.junet@uunet.uu.net
+81-3-234-5058

UNIX Fair '88 Association
1-1-1 Hirakawa-chu,
Chiyoda-ku, Tokyo 102
Japan

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

USENIX Association Office
P.O. Box 2299
Berkeley, CA 94710
USA
+1 415 528 8649
office@usenix.uucp

Sun UK User Group
Sue Crozier
Sun Microsystems, UK
+44 276 62111

EUUG National group addresses can be found on the back cover of this newsletter.

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

Everyone who comes here wants three things:

1. They want it quick.

2. They want it good.

3. They want it cheap.

I tell 'em to pick two and call me back.

Sign on the back wall of
a small printing company
in Delaware, USA

# Glossary

Here are definitions of a few of the not-so-common English words that can be found in this newsletter. Where a word has several meanings the way in which it is used in this issue is the one that is explained.

| | |
|---|---|
| accommodate | allow for/possible |
| achieved | successfully done |
| adequate | good enough |
| adopted | taken up/done |
| amaze | surprise |
| anomalies | inconsistancies/differences |
| anxious | eager to ... |
| aspect | part of/detail |
| authentic | verification/proof that it is what it claims to be |
| bogus | impostor/not the real one |
| capitulate | give in/admit defeat |
| cascaded | joined one after the other |
| ceases | stops |
| coincided | was the same as/overlapped with |
| consisted | made up of |
| contrary | different to/as opposed to |
| defamatory | polite/not rude |
| defer | leave to a later date |
| decree | law/statement by 'official' |
| deposited | put into ... |
| despaired | given up hope |
| devoted | given over entirely to |
| diminished | reduced/made smaller |
| disdain | turn your nose up at ... |
| disposal | for (your) use |
| encounter | meet/come across |
| enlighten | teach new fact |
| ensures | make sure that ... |
| esoteric | difficult to understand |
| forge | imitate/pretend to be something else |
| immersed | spend a lot of time doing ... |
| manipulation | handling and changing |
| mnemonic | easy to remember |
| nepotism | favouritism to friend/relative |
| obliged | made (by someone) to do ... |
| pathological | situation that causes a problem |
| prior | before |
| resumed | start again |
| snark | boojum (see Alice though the Looking Glass) |
| suffices | is enough to ... |
| tame | make a wild animal obey you |
| unwitting | do something without realising it |

# UKUUG Summer Abstracts

Here are the abstracts of the papers that will be delivered at the UKUUG meeting at Strathclyde on 27th-28th June 1989.

*Interconnection in a Graphical Environment*
*Robin Faichney - University of Kent, Canterbury*

One of the most significant features of UNIX is the ability to build applications from small, modular, reusable units. This means ease of inter-connection, achieved largely through command line redirection. The redirection of user-interaction is complicated in a graphical environment due to the nature and variety of graphical actions, and also the highly interactive nature of many graphics-interfaced applications. Two systems have been designed and implemented to facilitate interconnection. (While one is complete, the other exists as a partial prototype.)

The first is, in effect, a high level interface to existing inter-process communication facilities, incorporating additional features. The other consists of a library of routines designed to allow the redirection of all traffic between the user-interface and the underlying functionality of an application, to and/or from another program. Apart from the switching on/off of redirection, it is fully transparent to the application. Applications which demonstrate the use of both systems have been implemented.

*An Introduction to GoodNeWS*
*Arthur van Hoff - Turing Institute, Glasgow*

GoodNeWS is a NeWS-based window interface that was developed at the Turing Institute. It provides a complete windowing environment which includes terminal emulators, graphics tools and a LaTeX previewer. Most of the Sun-supplied PostScript software has been replaced to provide a more consistent interface. GoodNeWS also incorporates an object-oriented programming environment in PostScript called HyperNeWS, which presents a HyperCard-style user interface.

*An Interactive UNIX spelling corrector*
*Philip M Sleat, Sunil K Das - City University, London*

When designing interactive information processing applications there are two important properties of the data files thatmust be considered. The first of these is the size of the file. The second is the speed of access. An investigation was conducted into the most appropriate structures for the storage of dictionaries. Sequential, binary and hashing techniques proved to be inferior to tree-based methods which permitted data compression.

*An Overview of the Rekursiv*
*Bruno Beloff - Linn Smart Computing Ltd., Glasgow*

The Rekursiv is a novel computer architecture developed by Linn to provide a processor that efficiently executes object-oriented programming languages. This paper describes that architecture of the Rekursiv and discusses some of the problems of using UNIX as a "slave" operating system to the Rekursiv processor.

*NFS refreshes the filesystem(s) A/UX cannot reach*
*William Roberts,*
*Matthew Kempthorne-Ley-Edwards,*
*Robert Bradshaw*
*Queen Mary College, London*

A/UX, the Apple Macintosh version of UNIX, coexists alongside the original MacOS operating system: separate partitions on the same disk with little real connection between the two. The potential for running MacOS applications under A/UX lead us to develop A/UX software which provides access to the MacOS files using standard UNIX file handling. This paper describes the A/UX environment and the techniques we used: a first version based on a library to simulate UNIX file system calls and the later version using the NFS protocol to mount the MacOS partition as a UNIX filestore. We offer some reflections on the problems and successes of our work and suggest a number of things which may help those providing NFS servers for other 'exotic' filestores.

*The Independent—It Is. Are UNIX?*
*Dwight A. Ernest, Maria Tuck*
*Newspaper Publishing PLC, London*

The Independent is a major national quality newspaper in the UK. Since shortly after its launch in October 1986, its I.T. team has been using Sun workstations running a variant of the UNIX operating system to perform a wide and varied set of tasks. The importance and prominence of this operating system is increasing at the newspaper because of its 'developer friendliness', its standard networking capabilities, and the increasing price/performance ratio of various computers and workstations which are based on it (including Suns).

The presentation and accompanying paper will explain the environments in the development and operations area, and will describe how the various elements are networked. The use of the various other non-UNIX computers at the newspaper will also be very briefly explained so as to demonstrate how it all fits together and how UNIX networking provides indispensible links between the various disparate elements.

*UKnet Update*
*Steve Binns - University of Kent, Canterbury*

The head of the UKC Software Division, will give a short explanation of what UKnet is, how to join, the costs and other general information. He will also explain some of the future connectivity services both in the UK, Europe and the wider world for UKnet sites.

*Installing Software on UNIX*
*Tony Bamford - Parliament Hill Computers, Caversham*

How is a 'shrink-wrapped UNIX product' installed on any random UNIX engine? Traditional mathods are discussed, current standards work is looked at and possible 'great leap forwards' talked about. The presentation will follow up the issues raised from the recent London UNIX User Group meeting on this topic.

*Generis -*
*An Intelligent Knowledge-Based Management System*
*Jim Williamson - Deductive Systems Ltd., Glasgow*

Generis provides an intelligent knowledge engineering environment based on entity relationship database technology. This has been extended with object-oriented representation within a semantic network structure utilising the unique generic relational data model. Generis incorporates a comprehensive rule base together with integral spreadsheet, document management and intelligent query language.

*A Model-based Diagnostic System for UNIX*
*Gail Anderson - AIAI, University of Edinburgh*

A model of the UNIX Operating System based on file system structure is developed. Application of the model is described and current work on the project is outlined.

*Experience with a Graphics Processor as an X-Server*
*Stephen McInnes - Silicon Products Ltd., Glasgow*

This paper describes the problems encountered by the author in using a novel graphics processor to provide a high performance X-Windows server. This server will allow up to eight users to run graphics application programs under the X Window system. One application area will briefly be described—the system's use as a VLSI design tool.

*So You Think Your Network Is Slow?*
*William Armitage - University of Nottingham*

Phil Karn (KA9Q) has written a C based implementation of TCP/IP for packet radio applications that runs on a number of cheap machines, e.g., PC clones, Amigas. This code is public domain. As well as bringing familiar services to these small machines it is a useful network gateway for SLIP and ethernets. A number of the 4.3Tahoe network congestion handling enhancements arose from experience gained with this package in the much lower bandwidth environment of packet radio. This paper examines the package, its novel solutions to certain problems and some applications.

AUSTRIA - UUGA
Friedrich Kofler
✓ Austro Olivetti Ges.m.b.H
Braunhuberdasse 26
AUSTRIA
+43 222 743575 Ext 680
uuga@tuvie.at

FINLAND - FUUG
Ulla Patrikka - Jokinen
✓ PO Box 11
SF-02721 espoo
FINLAND
+358-0-509 1496
fuug-hallitus@fuug.fi

ICELAND - ICEUUG
Marius Olafsson
University Computer Center
✓ Dunhaga 5
Reykjavik
ICELAND
+354 1 694747
marius@rhi.hi.is

NETHERLANDS - NLUUG
Patricia Otter
✓ Xirion bv
World Trade Centre
Strawinskylaan 1135
1077 XX Amsterdam
THE NETHERLANDS
+31 20 6649411
nluug@hp4nl

UNITED KINGDOM - UKUUG
Bill Barrett
Owles Hall
✓ Buntingford
Hertfordshire SG9 9PL
UNITED KINGDOM
+44 763 73039
ukuug@ukc.ac.uk

The European UNIX systems User Group
Owles Hall
Buntingford
✓ Hertfordshire SG9 9PL
UK
+44 763 73039
euug@inset.co.uk

BELGIUM - BUUG
Marc Nyssen
✓ Department of Medical Informatics
VUB, Laarbeeklaan 103
B-1090 Brussels
BELGIUM
+32 2 477 4111
buug@vub.uucp

FRANCE - AFUU
Miss Ann Garnery
AFUU
✓ 11 rue Carnot
94270 Le Kremlin Bicetre
Paris
FRANCE
+33 1 46 70 95 90
anne@inria.inria.fr

IRELAND - IUUG
Norman Hull
Irish Unix-systems User Group
✓ Court Place
Carlow
IRELAND
+353 503-31745
norman@q2rs.ie

NORWAY - NUUG
Jan Brandt Jensen
Unisoft A.S.
✓ Enebakkvn 154
N-0680 Oslo 6
NORWAY
+47 2 688970

PORTUGAL - PUUG
Prof Legatheaux Martens
Avenue 24 de Julho no 134 7° andar
✓ Lisboa
Portugal
+351 1 668773
jalm@iuesc.pt

DENMARK - DKUUG
Mogens Buhelt
✓ Kabbelejevej 27B
DK-2700 Bronshoj
DENMARK
+45 1 60 66 80
mogens@dkuug.dk

HUNGARY - HUUG
Dr Elôd Knuth
Computer and Automation Institute
✓ Hungarian Academy of Sciences
H-1502 Budapest 112, P.O. Box 63
HUNGARY
+36 1 665 435

ITALY - i2u
Ing Carlo Mortarino
i2u
Viale Monza 347
✓ 20126 Milano
ITALY
+32 2 2520 2530

EUUG-S
N
I
X

SWEDEN - EUUG-S
Hans E. Johansson
✓ NCR Svenska AB
Box 1206
S-164 28 KISTA
SWEDEN
+46 8 750 26 03
hans@ncr.se

GERMANY - GUUG
✓ Dr Anton Gerold
GUUG-Vorstand
Elsenheimerstr 43
D-8000 MUNCHEN 21
WEST GERMANY
+49 89 570 7697
gerold@lan.informatik.tu-muenchem.dbp.de

YUGOSLAVIA - YUUG
Milan Palian
Iskra Delta Computers
✓ Parmova 41
61000 Ljubljana
Jugoslavija
+38 61 574 554
mpalian@idc.idcyuug.uucp