

NEWSLETTER

EUUG

European UNIX[®] systems User Group



Volume 7, No. 4
1987

CONTENTS

- Rule Based Systems in C++
- EUUG Strategy Workshop
- What's Up With EUUG?
- Fun With Spaces in Troff
- X/OPEN Mid-Term Report

EUROPEAN UNIX SYSTEMS USER GROUP NEWSLETTER

*Volume 7
Number 4
Winter 1987*

Editorial	1
EUUG Strategy Workshop	3
Rule Based Systems in C++	12
What NeWS?	36
The X/OPEN Show Revisited	42
UNIX Publications	44
The EUUG Conference at Dublin	51
EUUG Spring 1988 Conference	56
What's up with EUUG	60
Fun with Spaces in Troff	63
C++ Gossip	68
The X/OPEN Mid-Term Report	71
EUnet	73
UNIX in the Marketplace	77
EUUG Tape Distributions	81
News from Denmark	90
EUUGN Dublin Abstracts	92
Glossary	99
Conference Announcements	67, 100

This newsletter was set in Times Roman on a LaserGrafix LG800 printer, driven by troff from the UNIX time-sharing system.

Editorial Team:

Peter Collinson University of Kent at Canterbury
 (EUUG Secretary/Newsletter Editor)

Alain Williams Parliament Hill Computers

Sally Rutter The Instruction Set

Typesetting:

Sally Rutter The Instruction Set
Laura Dekker The Instruction Set
 City House,
 190, City Road
 London EC1V 2QH
 U.K.

Printed by: Atterbury Associates
 2 Passey Place
 London SE9 1BN
 U.K.

Published by: The EUUG
 Owles Hall,
 Buntingford
 Essex SG9 9PL
 U.K.

euug@inset.co.uk (...lmcvaxlukclinset!euug)

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

ISSN 1011-4211

Editorial

Alain D D Williams
...lmcvaxlukclinsetlphcompladdw, addw@phcomp

Parliament Hill Computers
London, England

1. *What have we been up to ?*

There have been two main events in the EUUG world recently. The first, a strategy meeting in Paris in September, was closely followed by the the 10th anniversary conference (part 1) in Dublin two weeks later. There are reports from both of these events in this issue.

In addition to Joe's report on the Guinness in Dublin I have included abstracts from the papers delivered at the conference. I hope that you find this useful and would like to see this extended to cover national group conferences.

The strategy meeting was the EUUG asking itself questions such as: "We are ten years old, what are we doing, what are we here for?" After much discussion there are still many questions that remain to be answered. Readers are encouraged to read the report and give their opinions. A small number of individuals make most of the noise in the EUUG, but they are *your* representatives and are interested in understanding your views.

2. *Multiple Newsletter Copies for Institutions*

Institutional members can now buy extra copies of the newsletter. This is intended to help where the newsletter is on a circulation list and it gets stuck on someones desk, or, in a large organisation, there is only one copy but there are several offices.

At the time of going to press the final details had not been worked out, but the price is likely to near the cost of production and all the copies will have to go to one address. Please contact your national group or Owles Hall for more information.

3. *Is your Postbag Big Enough ?*

The EUUG is updating its information mailing list. This is used to tell you about EUUG events like conferences. If you do not get your own copy of the newsletter then you ought to be on this list. Contact Owles Hall to ensure that you will be a victim of EUUG junk mail!

4. *Future Issues of the Newsletter*

Readers are encouraged to submit papers to the newsletter. The topics that I am interested in for the next two issues are:

— Security

How to protect data from unauthorised use or alteration, and how to stop theft of programs.

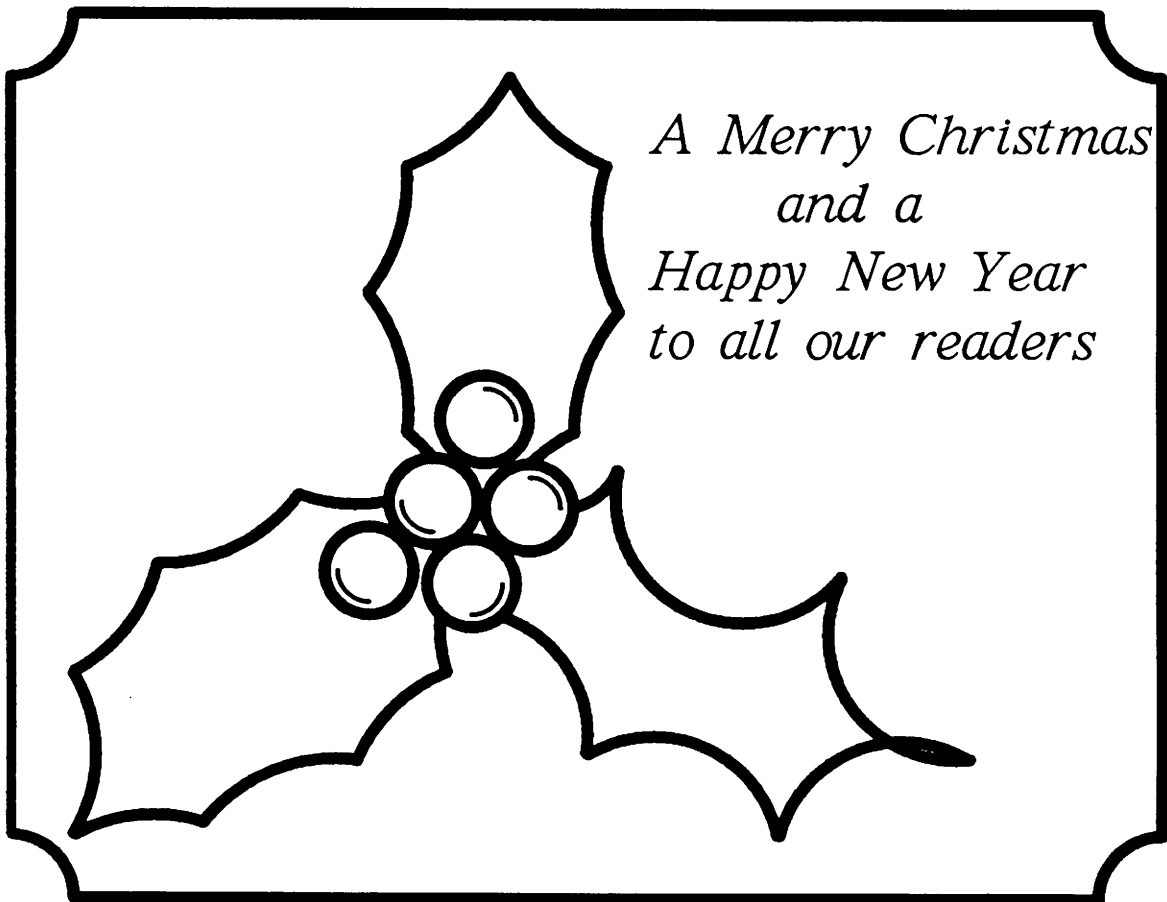
— Graphics

Presenting information in a pictorial way can often make it much easier to

understand. I am looking for discussions of the problems involved as well as case examples.

The next copy dates are:

25 January for 1 March
25 April for 1 June





EUUG Strategy Workshop
Paris 4-6 September 1987
Preliminary Notes

Helen Gibbons

euug@inset.co.uk, ...lmcvaxlukclinsetleuug

*EUUG Secretariat, Owles Hall
Buntingford, England*

1. A Weekend in Paris

This took place in the early September at the Chateau de Montvillargenne near Chantilly just north of Paris; an area well known for its horse breeding. This sounded like a nice relaxing idea, but ended up being jolly hard work. There was much discussion, some of it heated, as most people wanted their own particular ideas presented.

We shared the chateau with several wedding parties, some of whom played music very loudly under my bedroom window; I only got to sleep after the hotel manager gave me a bottle of Champagne to compensate for the noise!

Below are some rough notes that I took at the time:

The workshop dealt with the following topics: strategy, conferences and exhibitions, publications, the network, external relations, accounting and administration. Each of these will be dealt with under a separate heading.

2. Strategy

Drafting Committee Presenter: E Milgrom

This working party covered the role, purpose and organisation of the EUUG. It examined the role and purpose of the EUUG in respect to its members and considered whether individual group members should automatically be members of the EUUG or whether the EUUG should simply be an umbrella body with the National Groups as the members — this was preferred. So National Groups could be billed for central services and they could arrange their own special classes of membership.

Voting rights were looked at and all schemes proposed were democratic. The majority were in favour of "one country — one vote".

It was felt that the EUUG should help National Groups to improve their visibility. The EUUG should take a more important role internationally.

National Groups should retain a large amount of freedom in respect to members but it was felt that increasing institutional and corporate members was important.

At present the EUUG was seen as a forum for technical people, but it was suggested that more effort should be made to include more end users and prospective users, as the market might be too limited if restricted to technical people only. The question was raised — should the EUUG provide a platform for commercial companies? Should services include stands, exhibitions, data bases, etc?

It was felt that a distinction should be made between EUUG services and National Group services and the duration of each should be addressed periodically and financing for each service should be clearly identified.

Standards were seen as important, but enthusiasm was tempered by the amount of work needed. The majority thought that there were not adequate enough resources to fully take part in committee work, but they did want to promote standards.

Help services and duplication of efforts were looked at, and it was agreed that the EUUG should help the National Groups provide information to their members — i.e. tutorial materials, speakers, etc. and avoid conflicts of events wherever possible.

The EUUG should maintain a data base of useful information.

The EUUG should be involved in EUnet and provide support for EUnet by publicising and co-ordinating its activities. The majority agreed that it should fund studies and experiments, but *it should not run it*. The EUUG should, however, liaise with other bodies in Europe where appropriate.

Costs should be examined to see if they could be reduced without reducing services. Other means of income should be investigated.

Thought should be given to the importance of attendees at conferences, and the possibility of having other registered users within a corporation. Thought should be given as to what could be done for students.

There was a strong feeling that the Strategy Meetings were useful and should be continued, possibly every two years.

3. *Conferences and Exhibitions*

Drafting Committee Presenter: E. Janich

The current responsibilities were identified as:

Keld Simonsen	Conference officer
Helen Gibbons	Admin/Central office work
Niel Todd	Tutorials
Programme Committee	

At present the EUUG runs two conferences a year, and the other groups run from nought to two conferences per year. The profit varied from one event to the other.

The proposals of this working group were to settle on two EUUG conferences per year, one more commercial one in the Spring, and one more academic/technical one in the Autumn, but this should not be called a workshop.

The themes should be set well beforehand, and the call for papers should be distributed one year in advance. Time schedules should be set.

Tutorials should not be free. They should be more technical, and have a higher fee for non members but there should be no different levels (i.e. for profit and non profit makers).

The working group strongly recommended that there be a fixed and constant split between the host group for a conference and the EUUG, and that it be at the rate of 30/70 for all conferences and for a fixed amount of responsibilities, which should also clearly be identified.

Long term planning was also strongly recommended and it was suggested that the conferences should always be held one week after Easter (i.e. the first full week

after the Easter holiday week) and the last full week in September. This should be clear and constant and therefore should prevent collisions with group events, as it was felt to be important that EUUG conferences did not clash with local conferences. Neither should local conferences conflict with each other if they are in countries which speak the same language. Groups should talk to each other *before* fixing dates.

It was recommended that there be reduced fees for other national group members attending national conferences and this should be added to the official guidelines. The EUUG should co-ordinate invited speakers where-ever possible.

Exhibitions should be held solely for profit and should only be one per year for the EUUG.

As far as hiring professional organisers was concerned, it was felt that this should be the decision of the local group, but it should be under the control of the EUUG. Groups, however, were warned to be very careful and to keep a history of the experience, as there were unscrupulous organisers, and serious problems have been experienced in the past in using them. Care should be taken to use someone trustworthy.

It was felt that payment could be made for help when there was a heavy load a local organiser, so that local organisers could pay for jobs to be done or could offer in return free attendance at that and the next conference.

Local Groups should advertise their events and EUUG events in their local press.

It was thought that Usenix conferences had some good ideas and efforts should be made to learn from them.

Efforts should be made to advertise EUUG conferences in USENIX and also in Australia and Japan.

4. Publications

Drafting Committee Presenter: F. Kofler

The working party had examined:

Newsletter	Purpose	EUUG	Contents
National Groups	Production	Magazines	Finance
Technical Journal	Distribution	Others	Language

The working group recommended that the EUUG newsletter should not carry pure product information, but should carry a summary of what happens on the net news. It should report on surveys, reviews of books, products and public domain software. There should be more technical information at all levels, but the most technical should be placed with an outside technical journal, possibly Usenix.

The newsletter should provide cohesion and foster group feelings using pictures and jokes, but they must be of the kind understood by all. "In jokes" should be discouraged.

There should be no introductory material, but hints, pointers and lists and there should be reports of local working groups and market reports (if cheap) also legal aspects (licencing, copyrights).

It had been felt in the past that the newsletter was too English, but this was improving, and a good spread of European coverage did depend on contributions from the national groups, of which more were now being received. However, it was

agreed that all articles should appear in English. Translation services would be provided as would a service to tidy up English from non native English speakers.

It was noted from the survey that most local groups had their own newsletter, except for U.K., Ireland, the Netherlands and Iceland. Most published about 3 to 4 issues per year. Most of the work was done by the editors themselves, with the exception of Germany who paid for contributions. It was agreed the EUUG would not pay for articles and that the mailing list would not be available for sale.

The EUUG newsletter cost about £5,000, which works out at about £2 per copy*.

Funding should be from central budget at one copy per member, but extra copies should be made available to members at production plus 10%. Further copies should also be sent to the National Groups and to libraries.

Advertising should be allowed up to one third of the total. It was not possible to bind in different advertisements for different countries, but it was possible to provide different inserts for different countries.

It was recommended that the newsletter remained at 4 issues per year for the moment until the situation was stable, and then think of increasing to 6 per year.

The working party recommended that individual mailing be maintained as it was nice for members to have something coming direct from the EUUG, and the savings from bulk shipping would be marginal and added to that, it was not certain that all groups would be able to send out newsletters on a regular and reliable basis. The problems, however, in getting up to date labels from the groups was noted, but this was gradually improving.

National Groups should send a copy of their newsletter to each of the other groups, as well as to Owles Hall, and members could subscribe to other Groups' newsletters.

As far as other publications were concerned it was agreed to try to get a regular EUUG column in an important magazine.

Co-operation with Usenix was recommended, but the EUUG should not at present try to set up its own technical journal, nor should it try to produce its own catalogues, although a catalogue of catalogues would be useful and could be undertaken.

5. *The Network*

Drafting Committee Presenter: S. Kenyon

Current developments are that the EUUG has been talking to EEC about the present position of EUnet and its future. EEC interest is based on intensive use of EUnet for ESPRIT projects. A study has been proposed and is underway to investigate transition to X.400 and related ISO protocols. The results of Phase A will be presented at the Dublin conference. Phase B, which is to be completed by the end of 1987 if the EEC sign the contract, will look at migration. The EUUG has stipulated that software developed for this purpose will be available at a nominal handling charge to members of the EUUG.

The working party has expressed concern at the big influence of the backbone managers on the services provided. Suggestions to solve this included having the

* Secretary's note: On examining latest figures and including editors fee this seems to be working out at nearer 10,000 UK pounds per issue.

backbone manger on the National Board, and having more than one backbone. A lot of confusion had arisen around the question of the ownership of the network. Some national groups apparently own the backbone and send out bills directly. Some countries have no control whatsoever on the backbone policies. The working group had voted 50/50 on the ownership of the network by the National Groups. A majority is in favour of the National Group co-ordinating the network activities. Membership of the local user group was voted for as a mandatory requirement for connection to EUnet.

It was noted that at present the EUUG pays £3000 to £4000 annually to support EUnet (travel, meeting costs, etc.). It was generally agreed that EUnet should be made financially independent of the EUUG by raising the rates. This requires centralised billing, a subject which was discussed, but no decision was reached. A advantage of central billing might be the avoidance of VAT charges.

The entire cost/accounting situation has not been resolved, but it was felt that it should be.

Several proposals for future plans were made to expand the services provided by EUnet. These include selective news delivery on request, distribution of news, indexes, name server, user server, increased speed/reliability/connectivity, source archive sites for public domain software.

This raised the point as to how those services should be funded (whether they should be included in the basic charge or charged extra), and about alternative software distribution media (cassettes, floppies).

It was recommended that new gateways should follow the example set by the DFN gateway in Germany and the CSNET gateway at CWI.

The question of EUnet not allowing commercial exploitation was raised. This, however, seems to conflict with the contracting of backbone duties to a commercial company, something which is considered by some countries.

An idea was presented to set up a database of publication lists, software lists, news extracts, etc. and exploit this commercially.

The need for a manual on how to set up a network connection was identified. Two aspects, organisational and technical, were identified. The EUUG might provide a master which could be localised by every national group, but no commitment to create anything in this area was made.

It was reported that a public domain version of the UUCP software is now available. Doubts about the p.d. character were raised and ACSNET was mentioned as an alternative.

It was considered unwise to advertise the network heavily before legal problems with the PTTs have been resolved. EUnet should co-operate with its users and therefore not exclude PC users provided they are EUUG members.

It was felt that the results of the ECC study should answer some of the questions raised above, and that this report should be used as input to this study, together with any conclusions later added. The question was raised as to whether the study should consult with backbones, national groups, or both.

6. *External Relations*

Drafting Committee Presenter: I. Korn

A survey of the interactions between the national groups showed that most groups worked in isolation.

The working party examined what the groups were doing in relation to contact with other groups, contact with /usr/grp and Usenix, contact with vendors and with the press.

It seemed that a lot of working groups existed within the national groups and it was felt that the boards of the various groups and the Governing Board should look at how the knowledge of what they were doing could be spread.

There were, of course, language related problems, which would mean that regions with the same language were more likely to get together.

The EUUG should help to advertise events through, for example, the calendar and a limited mailing list, but in the end it would depend on the National Groups to advertise their events.

It was recommended that the Business Managers should call all the National Groups once or twice per month to maintain contact by asking what they were doing and what was new.

It was also recommended that the Secretariat be equipped and educated in the efficient use of email and that it should always respond to email messages.

The calendar, newsletter and events should be continued with extra effort.

To promote EUUG visibility there should be support for involvement with outside bodies such as EEC and CEPT and with standard bodies and committees such as ISO and X/OPEN.

To encourage vendor awareness National Groups should always include the words "affiliated to the EUUG" in their advertising and publicity, and vice versa — the EUUG should always stress its 14 groups.

UNIX training houses should be encouraged to promote the EUUG and relevant National Groups by distributing literature and information.

7. Accounting and Administration

Drafting Committee presenter: N. Todd

At present Helen Gibbons is paid by the EUUG to provide a secretariat service which includes the provision of staff, equipment, premises, heating, lighting, etc. involved in the running of an office. For a fixed fee all non conference activities are funded centrally while activities related to conferences are funded from conference budgets. Principal members of the Owles Hall staff are:

1. The Business Manager who represents the Secretariat at the Governing Board and Executive Committee meetings, deals with all administration, checks out proposed conference venues and handles paperwork associated with conferences, supervises financial aspect and prepares the financial reports. She is the "public face" of the Secretariat.
2. The Accountant, who is qualified under U.K. law. She works part time and keeps all financial records, prepares reports for the treasurer and assists the external auditors
3. A small number of secretarial staff are available to handle photocopying, typing, telephones, packet preparation etc. They also deal with the franking machine, or postage stamps for shorter jobs where the benefit of the franking

machine is outweighed by the trouble of resetting the machine for different rates of small runs.

Further details of Owles Hall can be found in the September 1987 newsletter.

The existing computing facilities at Owles Hall are currently a terminal and modem to INSET in London. The modem has recently been upgraded to 2400 band to increase reliability. Other than that it should be noted that at present the Secretariat has no computing facilities, though the EUUG working party on this is still looking into the provision of a machine. Currently favoured is a 386 based system to provide the opportunity to run DOS or UNIX. The working party recommended that sponsorship be sought and suggestions, comments, contact names or offers should be submitted to the computer working party via Owles Hall.

On financial policy and cash flow, it was noted that the current policy is to cover 6 months expenditure including a conference.

In reality there is currently about one years operating held in reserve, although this does not take into account a number of extraordinary items of expenditure such as the strategy workshop, the EEC networking study, payment of a newsletter editor, 10th anniversary expenditure, and additional executive committee meetings.

The Executive Committee monitors the financial position by means of the financial overviews prepared for the Treasurer by the accountant at Owles Hall. The Governing Board receives these via the Executive Committee minutes, plus a special set usually prepared to coincide with Governing Board meetings.

A two year financial projection is prepared by the treasurer, and there was general agreement that this time scale was sufficient.

On subscriptions, points 5.2 and 5.3 of the current bye laws (which are now being revised) were re-iterated, and the working group requested that the invoices sent to the National Groups should have an explicit "pay by" date and that full bank details should be included to facilitate payments. The working group also suggested that the use of Giro bank accounts be investigated, and that there might be an advantage of looking into the possibility of the EUUG setting up "off shore" accounts.

At present a visa card had been obtained for the use of the Business Manager for official EUUG business, and Barclaycard were currently investigating the EUUG before deciding whether to allow it to take payment by credit card, but it was hoped that this would be resolved soon.

It was requested that the National Groups check the membership details sent to Owles Hall to ensure that extra names are not included by mistake.

It was noted that there have been problems in the past in trying to reclaim VAT from other countries, such as France. Care should be taken to avoid VAT problems as much as possible by using local merchandise.

Financial transactions took up a great deal of the time of the Secretariat. Members were asked to help by trying to ensure that expense claims were not submitted in multiple currencies and that the currency itself was always made clear.

Current policy required the National Groups to submit a copy of their accounts to Owles Hall for distribution to the other national groups. The working group felt that there was little to be gained by this and that a National Group's financial affairs were their own concern. Also the effort of converting the various accounts into a comparable form was too large in relation to the benefit. The working group recommended that this practice be discontinued.

It was agreed that address labels should always be requested by email and realistic reminders and deadlines should be used.

It was agreed that the Secretariat needed a higher profile on the network. Future provision should be made for the electronic update of membership lists.

EUUG/National Group co-operation and liason was seen as important and, finance permitting, it was suggested that the Business Manager should visit the committees of the National Groups. A slide presentation about the EUUG was already being considered and this could be used by any member of the committee visiting existing and start up groups.

The working group had discussed the membership cards and felt that they had been a failure. It was recommended that the scheme be abandoned.

The wall chart was to be repeated for 1988 and corrections and details from the National Groups should be sent to Owles Hall by 1st November 1987.

The Business Manager invited suggestions for the updating of the bye-laws and the membership leaflet and asked that they should be sent directly to her.

Finally, it was noted that the administration was improving and that the Secretariat provided a prompt and reliable service.

General Discussion

At the end of the presentations there followed a general discussion on policy.

A special working party on fees reported their discussions. The fee to the EUUG should depend on the services provided. It was noted that it was not possible to lower fees *and* raise services, but the student question should be addressed. It was agreed that all members of national groups should be EUUG members.

The Executive Board Structure was discussed. It was thought that more work should be done on a national level. The Executive committee was responsible for ensuring that the work was done, but it did not need actually to do the work. Items to be addressed within the executive committee were: finances, publications, conferences, tutorials, software, EUnet, PR, Internationalisation User Group Relations, Administration, and co-ordination.

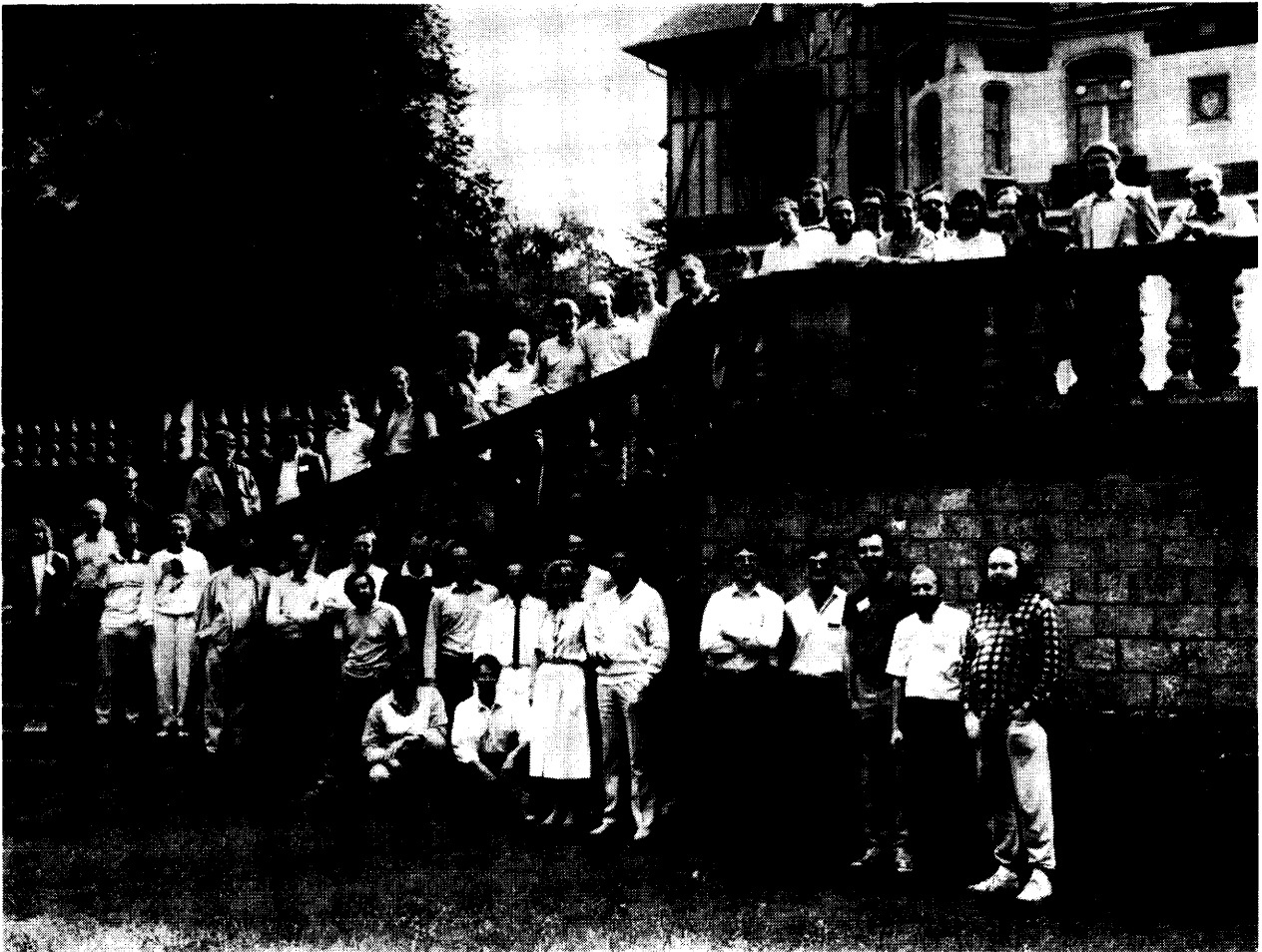
Some thought should be given to whether people should be elected to the executive committee and then placed in the jobs, or whether people should be chosen to do a specific job. It was accepted that it was difficult for the National Groups to vote as they did not know the people. It was agreed, therefore, that everyone standing for the executive committee should produce a position paper about himself/herself.

It was agreed that thought should be given to new ideas and that a shift in services should be considered towards providing less existing services and more new ones.

A suggestion was that the EUUG newsletter should become thinner while despatching information centrally to the National Groups for them to include in their newsletters.

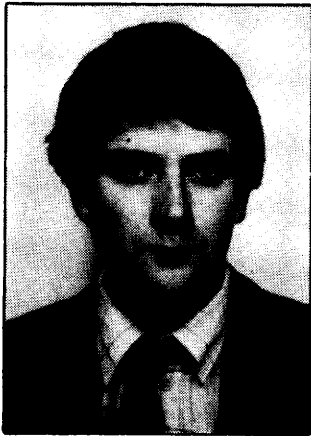
It was agreed that one Governing Board meeting each year should be separated from the conference so that more time could be given to discussions.

If you would like comment on any part of this then please send your thoughts to Owles Hall, or email to euug-exec@cw.nl (...!mcvax!euug-exec), or to your national group representatives.



The Morning After the Paris Strategy Meeting

A Preprocessor Extending C++ to Support Rule Based Systems via Access Orientation



Daniel Mann
dan@udi.uucp

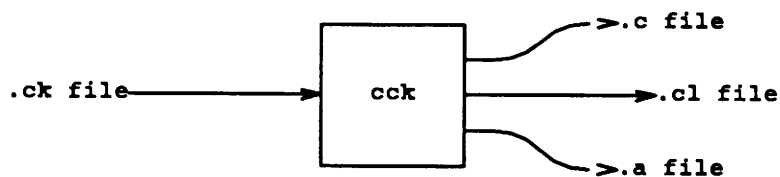
UDI Group Limited
Aberdeen, Scotland

Daniel Mann is Senior Systems Analyst at UDI Ltd. Previously he has worked in both hardware and software development environments. UDI is involved in designing and manufacturing underwater sonar equipment. The work described here was used in prototyping a sonar security system. He is currently investigating the use of UNIX in real-time control applications.

Introduction

These notes describe an experimental extension to C++ developed at UDI by Daniel Mann and Graham Rowbotham. The system is dealt with in some detail and the reader is required to have an understanding of C++ and rule based systems such as OPS5. The notes are intended to give a brief description of the implementation and its potential. A detailed knowledge of the implementation is not necessary to make effective use of the the preprocessor. Some readers may prefer to study the examples given in the Appendix before reading the main text.

The `cck` preprocessor enables rule constructs to be incorporated in C++ code. The preprocessor converts the effectively extended language statements into C++ statements. Any existing C++ code is unaffected. Source files which contain rule statements have the file name postfix `.ck` rather than `.c`. File names ending in `.ck` are passed through the `.cck` preprocessor before being passed to `cfront`. When `.ck` files are compiled with the `CC` command three files may be produced by `cck`. There is always a `.c` file which contains regular C++ code. There may be `.a` and `.cl` files. The `.a` file contains a synopsis of the symbol objects accessed by rules. The `.cl` file contains declarations of symbol objects and rule objects defined in in the `.ck` file.



A Simple Example

The rule statements can best be described by a simple example. The rules given are derived from the "Robbie spends a day at the Zoo" example (p177, *Artificial Intelligence II*, Patrick Henry Winston). The `carnivore_1` rule has two left hand side tests (if parts). If all of the LHS test conditions are satisfied then a rule is said to be triggered, and the code in the rule right hand side, RHS, (then part) is scheduled for execution (firing).

```
#define CMP(lst, rst) !strcmp(lst, rst)

rule carnivore_1
{
    ( CMP(animal.@is, "mammal") );
    ( CMP(observe.@eats, "meat") );
-->
    printf("it is a carnivore\n");
    animal.@is = "carnivore";
}

```

Rule RHS consist of any executable C++ statements. Rule LHS consist of a number of arbitrary logical expressions operating on working memory objects and other data. The logical expressions are arranged in a number of conditional tests. Each test is known as a conditional element (as in OPS5). The two conditional elements in the `carnivore_1` rule make use of the `CMP` macro to test if object member text pointers match with literal strings. The macro is used for convenience only. Let's review the first conditional element above.

```
( CMP(animal.@is, "mammal") );
( <expression> ); // in general

```

A conditional element can contain any valid C++ expression which can be evaluated to "true" or "false". The function `strcmp` is used to test if the string pointed to by object `animal` member `is` matches with literal string "mammal". The only unusual feature of this expression is the `@` character. It is used in rule LHS to indicate that the rule is "monitoring" the object field. In general instances of:

```
<object>.@<member>
```

appearing in a rule LHS are identified by the `cck` preprocessor which responds by removing the `@` character before passing the remaining C++ code to the output `.c` file. Also, an entry is made in the `.a` file for the `carnivore_1` rule:

```
animal_wmo      is      carnivore_1    ce_0
observe_wmo     eats    carnivore_1    ce_1

<object class> <member> <rule>      <conditional element>
```


The .a file contains information required to build an attached function for the object members monitored by the rules. Attached functions contain calls to rule conditional element test functions. Member functions which appear in rule LHS must have an attached function declared in the object class definition. Consider the animal object used above:

```
class animal_wmo
{public:
    char*   is;      // member
    void    is_a(); // attached function for 'is'
}

animal_wmo    animal; // instance of animal_wmo class
```

As far as rules are concerned there is only one instance of a working memory object. The animal object is assumed by the rules to be of class animal_wmo. Rule statements are converted into rule objects of which there is only one instance. The rule objects have member functions which perform the conditional element tests and RHS statements. When the attached function is_a() is called for an object of class animal_wmo (that is the animal object) a message is sent to the appropriate rule objects causing them to evaluate the affected conditional elements. The is_a() function will look something like:

```
void animal::is_a()
{
    carnivore_1_rule.ce_0();
    // other rules monitoring animal.is
}
```

Rule RHS statements follow the --> token. Alternatively the -fire-> token can be used. Rule RHS consist of any executable C++ statements. Normally RHS code assert new object member values. For example, the rule carnivore_1, when fired, updates the "animalis" member with the statement:

```
animal.@is = "carnivore";

<object>.@<member>      // lvalue
```

When the @ operator is used in the rule RHS, or in executable C++ code outside rules, the preprocessor inserts a call to the attached function, thus informing rule LHS of the value change. The above assignment results in the following C++ code being inserted into the output .cfile:

```
animal.is = "carnivore"; // assert new value
animal.is_a();           // call attached function
```

The Class Definition file, .cl

Rule statements produce rule objects. These objects have functions which perform the conditional element tests and rule RHS. The member functions are output to the .c file. The rule class declarations are output to the .cl file. This enables other rules to monitor objects defined in files other than the file containing the rule statement. Each file using any particular rule must `#include` the .cl file associated with the .ck file defining the rule. Let's look at the `carnivore_1` rule:

```
rule carnivore_1
{
    ( CMP(animal.@is, "mammal") );
    ( CMP(observe.@eats, "meat") );
-->
    printf("it is a carnivore\n");
    animal.@is = "carnivore";
}
```

The .c file shall contain the instance of the rule class and the definition of the member functions. The listing below shows part of the code output to the .c file:

```
carnivore_1_rule      carnivore_1(3); // rule instance

void carnivore_1_rule::ce_0() // conditional element
{
    int cetest = CMP(animal.is, "mammal");
    testlhs(cetest, 0);
}
```

The value 3 passed to the `carnivore_1` constructor is used to initialise the conditional element test to all-false. The state of the conditional elements is stored by a bit-field representation in the `base_rule`, known as the summary value. The listing below shows the code output to the .cl file. This code defines the rule object.

```
class carnivore_1_rule : public base_rule
{public:
    carnivore_1_rule(int sum) : (sum){}
    void ce_0();
    void ce_1();
    void fire();
};
```

The wmo Statement

The `wmo` statement is used to declare working memory objects which are to be known in other files. The `wmo` statement below demonstrates the syntax:

```
wmo animal
{public:
    int     @mammal;
    int     @carnivore;
    int     @ungulate;
    char*   @is;
};
```

The operation of the `wmo` statement is very simple. The `wmo` token is replaced with the class keyword. The class name is given the extension `_wmo` and the whole of the class statement is output to the `.cl` file. Any members prefixed with the `@` operator have an attached member function included in the class declaration. The `.cl` output for the above example would be:

```
class animal_wmo
{public:
    int     mammal;           // members
    int     carnivore;
    int     ungulate;
    char*   is;
    void    mammal_a();      // attached functions
    void    carnivore_a();
    void    ungulate_a();
    void    is_a();
};
```

The `wmo` statement also causes a class instance to be inserted in the `.c` file. For the above example the `.c` file would contain:

```
animal_wmo    animal;
```

If the `wmo` has a constructor function, then initialisation values can be included in the `wmo` statement which will be inserted into the `.c` file class instance. Consider the example below:

```
wmo animal
{public:
    animal_wmo(int, char*);
    int     @mammal;
    int     @carnivore;
    int     @ungulate;
    char*   @is;
}(34, "frog");
```

Building Attached Functions

Each `.ck` file operated on by `cck` will produce a `.a` file if rule statements are used. The `.a` file contains the information needed to build the attached functions. The

attached functions are used to inform rules that objects appearing in their LHS have changed value.

When all the .ck files have been compiled there will be a number of .a files produced. These .a files must be operated on together to produce a single .c file which contains all the attached functions. Consider the two .a files given below.

```

animal_wmo      is      carnivore_1    ce_0 // file1.a
observe_wmo     eats    carnivore_1    ce_1

animal_wmo      is      tiger           ce_0 // file2.a
animal_wmo      is      ostrich          ce_0

```

The `abld` command is used to combine the .a files and build the attached function file. The following command will produce an attached function file named `afunc.c`:

```
abld -o afunc.c file1.a file2.a
```

The `afunc.c` file can then be compiled and linked with other C++ modules. The listing below shows the contents of the `afunc.c` file.

```

void  observe_wmo::eats_a() // attached Fn for observe.eats
{
    carnivore_1.ce_1();
}

void  animal_wme::is_a()    // attached Fn for animal.is
{
    carnivore_1.ce_0();
    tiger.ce_0();
    ostrich.ce_0();
}

```

The Base Rule

Each rule object is derived from a base rule. The base rule class is declared in the `engine.cl` file. This file along with the executable inference engine file, `engine.c`, is available for the programmer to modify. By adding additional base rule constructs it is possible to initialise the base rule members. The preprocessor passes all rule initialisation data to the rule object definition statement in the .c file. Consider the example rule below:


```

// initialised rule
rule carnivore_1(60)
{
    ( CMP(animal.@is, "mammal") );
    ( CMP(observe.@eats, "meat") );
-->
    animal.@is = "carnivore";
}

// resulting '.c' file rule object instance
carnivore_1_rule    carnivore_1(3, 60);

```

The constructor generated for rules which have initialisation uses the `base_rule_macro`. This macro is defined in `engine.cl` and must be modified to deal with the parameters passed to the base rule constructor. In the above example the value 60 is passed to a base rule member `strength`, which is used to give strength ordering to rules. The listing below illustrates the code output to the `.cl` file.

```

#define base_rule_macro (int sum, int str):(sum, str){ }

class carnivore_1_rule : public base_rule
{public:
    carnivore_1_rule base_rule_macro
    void ce_0();
    void ce_1();
    void fire();
};

```

All information about rules is stored in the base rule. Because the base rule is entirely under programmer control there is great scope for extending the information stored in rule objects. The listing below presents a possible base rule object.

```

class base_rule // from engine.cl file
{
    friend meta_rule; // only meta rule can access
    base_rule* rule_fp; // pnt to next rule in agenda
    base_rule* rule_bp; // pnt to previous rule in agenda
    int oldsum;
    int summary; // conditional element summary
    int triggered;
    int strength; // rule strength
    int priority; // rule priority
protected:
    float confidence; // by derived rule
public:
    base_rule(int sum){ summary = sum; }
    base_rule(int sum, int str)
        { summary = sum, strength = str;}
}

```

```
void    testlhs(int ,int );  
virtual void    trig(){  
virtual void    fire(){  
  
};
```

The Meta Rule

In the `engine.c` file an object known as the meta rule is defined. This object controls the firing order of the rules. When the conditional elements of a rule are all satisfied then the rule is triggered. This is done by calling `meta.trig()`. This function determines where the triggered rule object is to be inserted in an agenda of rules waiting to be fired. The function `meta.fire()` removes the top rule from the agenda and executes the rule RHS. By running `meta.fire()` until the agenda is empty, the "inference engine" applies the knowledge contained in the compiled rules to the working memory objects repeatedly, until no new object member values can be produced.

The programmer controls the rule firing order via the information stored in each rule object base part and the `meta.trig()` function. The listing below illustrates an example function:

```

// ***** meta_rule::trig
//      Function to insert a triggered rule in the agenda.
//      The agenda is ordered according to rule priority.
//      Only one instance of a given rule is allowed.
//
void  meta_rule::trig(base_rule* insert)
{
    base_rule* prev = 0;
    int      priority;

    // calculate rule priority-----
    priority = insert->strength * 100
              + insert->confidence;

    // If rule is already triggered with a different
    // priority then remove previous instance of
    // rule from the agenda.
    if (insert->triggered)
    {
        if (priority == insert->priority)
            return;
        else
            meta.remove(insert);
    }

    // add rule to agenda-----
    while (agenda && agenda->priority > priority)
    {
        // advance along agenda
        prev = agenda;
        agenda = agenda->rule_fp;
    }
    insert->triggered = 1; // flag rule as triggered

    ....
}

```

In the above example rules are added according to a priority value. This value is determined from the rule strength and confidence values. Rules are only triggered once for any particular LHS conditions (the refraction principle). If a rule already exists in the agenda at a priority level of the rule to be inserted, then the new rule is inserted ahead of the already waiting-to-fire rule (the recency principle). The order of rule firing is completely under the control of the programmer and need not follow the above example.

The -trig-> Statement

If the agenda ordering is dependent on the LHS values then it is necessary to examine the LHS objects before the meta rule inserts the triggered rule into the agenda. This is done with the `-trig->` statement. Any C++ code can be executed before the rule fires, at the time the rule is triggered. The example below illustrates the syntax:

```

rule carnivore_1
{
    ( CMP(animal.@is, "mammal") );
    ( CMP(observe.@eats, "meat") );
-trig->
    confidence = min(animal.conf, observe.conf);
-fire->
    animal.@conf = confidence;
    animal.@is = "carnivore";
}

```

The rule confidence member is evaluated before `meta.trig()` is called. The confidence member can then be used by `meta.trig()` to order the firing.

Syntactically Attached

The attached functions are determined at compile time. This leads to some interesting effects when using pointers to modify object members. It is not possible to modify the attached function for the object member dereferenced by a pointer. However, if the dereferenced object is modified via indirection, rule LHS explicitly monitoring the object shall not receive a message informing them of the change. In such cases it is necessary to use the indirection operator in the rule LHS. Consider the working memory object given below:

```

wmo power
{
    int    engine1;
    int    engine2;
public:
    int    maximum;
    int    *@value; // points to engine member
};

```

Only the value member has an attached function. It is used to point to either `engine1` or `engine2`. The necessary C++ code for initialising the members is not shown. If a rule is monitoring the power object for whichever engine is currently in use it must dereference the value member, as does the example below.

```

rule more_speed
{
    ( *(power.@value) < power.maximum );
-->
    (*power.@value)++;
}

```

The rule RHS increments the dereferenced engine member. It also calls the attached function for `power.value`. This is quite correct, but the example rule will cause a knowledge loop. As the rule RHS continually messages its own LHS until the engine value is incremented to `power.maximum` and the rule no longer triggers.

Non-Monotonic Reasoning

Non-monotonic reasoning is used to remove the effects of a rule RHS made on out-of-date LHS data. When an attached function is called, rule conditional element functions are evaluated and the result passed to the base rule function `base_rule.testlhs()`. When a rule previously fired experiences a change in a working memory object tested by the rule LHS, such that the rule no longer applies, the summary bit-field value is updated by the `testlhs()` function such as to mark the rule as not fired. The listing below presents the function. As usual the programmer can modify its operation to suit.

```
void    base_rule::testlhs(int cetest, int ceid)
{
    int    mask = 1 << ceid;
    summary = cetest ? (summary & ~mask)
                   : (summary | mask);

    if( !summary          // test if rule fired
        || meta.nonmon() // test for truth maintenance
        && !oldsum)
    {
        this->trig(); // execute rule -trig->
        meta.trig(this); // add rule to agenda
    }
}
```

The base rule has a "non-monotonic" member which if set will cause the rule to be fired when it is marked as not fired and is currently fired. In this way the expert rules can deal with altering conclusions in response to new data. Only conclusions drawn from currently valid data are maintained. Changes in conditions may well result in previously derived conclusions being deleted in order to maintain a "true state" for the information derived by the inference engine.

The Zoo Example

The complete code for the "Robbie spends a day at the Zoo" problem is given in Appendix A. The file `animal.ck` must be compiled with the CC command along with the C++ inference engine file, `engine.c`. The `-a` option indicates that `abld` should be run on each `.afile` generated by `.ckfiles`. The `abld` command builds a C++ file with default name `abld.c` which is linked in with the other modules to make an executable programme, `animal`.

```
CC -v -a animal.ck engine.c -o animal
```

The CC command executes `cck` with the option to insert line control information. This enables the compiler to report errors occurring in the original source `.ck` file. Also the `cdb` debugger reports line numbers from the `.ck` file rather than the translated `.c` file. To examine the output C++ code from the preprocessor without line control information, run `cck` separately with the `-P` option.

In the zoo example, two working memory objects are used; `observe` and `animal`. Only one is necessary, but two demonstrate better the operation of the preprocessor.

In the main() function four observe members are updated. After each new value assertion, the inference engine is run to derive new information. The listing below presents the screen output. The speed of the system has been measured on an HP model 550 mini at 500 logical inferences per second.

```

-----new animal
  I can see it eating meat
  I can see it has hair
its a mammal
it is a carnivore
  I can see it has dark spots
  I can see its colour is tawny
it is a cheetah

```

Backward Chaining

A version of the well known monkey and banana problem has been coded using the preprocessor with C++. The programme is listed in Appendix B. Working memory objects are used to represent: nothing, bananas, ladder, couch, floor, ceiling and the monkey itself. Most of these objects are derived from a base object. The reason for the large number of objects is the extensive type checking performed by C++. The compilation command for the problem is given below:

```
CC -v -a mab.ck mab_main.ck engine.c -o mab
```

The problem is solved using backward chaining. Working memory objects are used to store information about goals. This highlights a difficulty occurring in backward chaining systems. There can be only one instance of any particular goal active at any instant in time. To solve this problem a base goal object is used which has a stacking mechanism defined. Each goal object is derived from the base goal and inherits the stacking implementation. When a particular goal is active and another goal of the same type is required, the active goal is stacked until the new goal is satisfied. Unfortunately this makes it impossible to test, in rule LHS, multiple instances of goals of the same type.

The listing below presents the screen output produced by the programme execution. The initial conditions place the monkey on the couch and the bananas on the ceiling at position 3. Because of the stacking mechanism, some goals are satisfied twice. This reduces the effective speed of the system. The speed has been measured in an HP 550 mini at 428 effective logical inferences per second.

```

initial conditions are set
set up goal to hold bananas
mb1 goal_move object ladder to 3
mb8 goal_holds object ladder      < stack hold bananas
mb5 goal_walk_to ladder
mb11 goal_on floor
mb14 jump onto the floor

```

```

mb12 walk to ladder
mb7 grab ladder
  restore stacked goal          < restore hold bananas
mb1 goal_move object ladder to 3 < stack move ladder to 3
mb9 goal_walk_to 3
mb13 walk with ladder to 3
mb10 goal_move satisfied
  restore stacked goal          < restore move to 3
mb10 goal_move satisfied
mb6 goal_holds nothing         < stack hold bananas
mb18 drop ladder
  restore stacked goal          < restore hold bananas
mb2 goal_on object ladder
mb17 climb onto ladder
mb4 grab bananas

```

Conclusion

These notes report the work carried out in developing a C++ preprocessor which supports rule based systems. Because rule knowledge is compiled, the speed of execution is fast. The syntax of the rule statements is easily understood by a C++ user. By extending C++ a number of advantages are gained: rules can be added to existing C++ code; rule based techniques can be easily mixed with the techniques supported by C++; the rule firing order is under the control of the programmer, not built into the system. The system is easily portable to different execution environments.

The disadvantages of the implementation are: there is no interpretive development option; each modification to rule knowledge requires a re-compilation; backward chaining systems are restricted due to the need to have only one instance of a working memory goal object in use at any instant.

Appendix 1

```

// @(#)animal.ck 1.3 87/06/26 15:47:41
// compile this problem with:
//      CC -v -a animal.ck engine.c -o animal
#include "engine.cl"
#include "animal.cl"
#include "stdio.h"
#include "string.h"

#define CMP(lstr,rstr) !strcmp(lstr,rstr)
#define TRUE 1

extern meta_rule meta;

// ***** the symbols
wmo observe
{public:int      @hair;
  int           @gives_milk;
  int           @feathers;
  char*         @flies;

```

```

    int      @lays_eggs;
    char*    @eats;
    int      @pointed_teeth;
    int      @claws;
    char*    @eyes;
    int      @hoofs;
    int      @chews_cud;
    int      @even_toed;
    char*    @colour;
    char*    @stripes;
    char*    @legs;
    char*    @neck;
    char*    @spots;
    int      @swins;
};

wmo animal
{public:int      @mammal;
    int      @carnivore;
    int      @ungulate;
    char*    @is;
};

// ***** main loop
main()
{
    printf("----new animal \n");
    printf(" I can see it eating meat\n");
    observe.@eats = "meat";
    while(meta.fire());
    printf(" I can see it has hair\n");
    observe.@hair = TRUE;
    while(meta.fire());
    printf(" I can see it has dark spots\n");
    observe.@spots = "dark";
    while(meta.fire());
    printf(" I can see its colour is tawny\n");
    observe.@colour = "tawny";
    while(meta.fire());
}

// ***** the rules

rule mammals
{
    ( observe.@hair || observe.@gives_milk);
-->

    printf("its a mammal\n");
    animal.@is = "mammal";
}

rule bird_1
{
    ( observe.@feathers );
-->

    printf("its a bird\n");
}

```



```
        animal.@is = "bird";
    }

rule bird_2
{
    ( CMP( observe.@flies, "yes" ) );
    ( observe.@lays_eggs );
-->
    printf("its a bird\n");
    animal.@is = "bird";
}

rule carnivore_1
{
    ( CMP( animal.@is, "mammal" ) );
    ( CMP( observe.@eats, "meat" ) );
-->
    printf("it is a carnivore\n");
    animal.@is = "carnivore";
}

rule carnivore_2
{
    ( CMP( animal.@is, "mammal" ) );
    ( observe.@pointed_teeth );
    ( observe.@claws );
    ( CMP( observe.@eyes, "point_forward" ) );
-->
    printf("it is a carnivore\n");
    animal.@is = "carnivore";
}

rule ungulate_1
{
    ( CMP( animal.@is, "mammal" ) );
    ( observe.@hoofs );
-->
    printf("it is an ungulate\n");
    animal.@ungulate= TRUE;
}

rule ungulate_2
{
    ( CMP( animal.@is, "mammal" ) );
    ( observe.@chews_cud );
-->
    printf("it is an ungulate\n");
    animal.@ungulate = TRUE;
    observe.@even_toed = TRUE;
}

rule cheetah
{
    ( CMP( animal.@is, "carnivore" ) );
    ( CMP( observe.@colour, "tawny" ) );
    ( CMP( observe.@spots, "dark" ) );
-->
    printf("it is a cheetah\n");
    animal.@is = "cheetah";
}
```

```
rule tiger
{
    ( CMP(animal.@is, "carnivore") );
    ( CMP(observe.@colour, "tawny") );
    ( CMP(observe.@stripes, "black") );
-->

    printf("it is a tiger\n");
    animal.@is = "tiger";
}

rule giraffe
{
    ( animal.@ungulate );
    ( CMP(observe.@legs, "long") );
    ( CMP(observe.@neck, "long") );
    ( CMP(observe.@colour, "tawny") );
    ( CMP(observe.@spots, "dark") );
-->

    printf("it is a giraffe\n");
    animal.@is = "giraffe";
}

rule zebra
{
    ( animal.@ungulate );
    ( CMP(observe.@colour, "white") );
    ( CMP(observe.@stripes, "black") );
-->

    printf("it is a zebra\n");
    animal.@is = "zebra";
}

rule ostrich
{
    ( CMP(animal.@is, "bird") );
    ( CMP(observe.@flies, "not") );
    ( CMP(observe.@legs, "long") );
    ( CMP(observe.@neck, "long") );
    ( CMP(observe.@colour, "black_and_white") );
-->

    printf("it is an ostrich \n");
    animal.@is = "ostrich";
}

rule penguin
{
    ( CMP(animal.@is, "bird") );
    ( CMP(observe.@flies, "not") );
    ( observe.@swims );
    ( CMP(observe.@colour, "black_and_white") );
-->

    printf("it is a penguin\n");
    animal.@is = "penguin";
}

rule albatross
{
    ( CMP(animal.@is, "bird") );
    ( CMP(observe.@flies, "good") );
-->
```

```

        printf("it is an albatross\n");
        animal.@is = "albatross";
    }

```

Appendix 2

```

// @(#)mab_main.ck 1.2
#include "engine.cl"
#include "mab.cl"
#include "stdio.h"

extern meta_rule meta;

enum    { light_c, heavy_c, pos_5_7_c, pos_2_2_c, pos_9_5_c};

extern  nothing_wmo      nothing;
extern  bananas_wmo     bananas;
extern  ladder_wmoladder;
extern  couch_wmo      couch;
extern  floor_wmo      floor;
extern  ceiling_wmo    ceiling;
extern  monkey_wmomonkey;

extern  goal_holds_wmo  goal_holds;
extern  goal_walk_to_wmo goal_walk_to;
extern  goal_on_wmo     goal_on;
extern  goal_move_wmo   goal_move;

// ***** main
main()
{
    couch.at = pos_5_7_c;
    couch.on = &floor;
    couch.weight = heavy_c;

    bananas.on = &ceiling;
    bananas.at = pos_2_2_c;

    ladder.on = &floor;
    ladder.weight = light_c;
    ladder.@at = pos_9_5_c;

    monkey.@on = &couch;
    monkey.@at = pos_5_7_c;
    monkey.@holds = &nothing;

    goal_holds.activate(&bananas);
    printf("initial conditions are set\n");
    printf("set up goal to hold bananas\n");

    while( meta.fire() );      // start inference engine
}

```

```

// @(#)mab.ck      1.4 87/06/26 16:28:28
// to compile this problem enter:
//      CC -v -a mab.ck mab_main.ck engine.c -o mab
#include "engine.cl"
#include "mab.cl"
#include "stdio.h"

extern meta_rule meta;

enum      { light_c, heavy_c, pos_5_7_c, pos_2_2_c, pos_9_5_c};

// ***** the symbols
wmo object
{public:
    object_wmo(){ }
    object_wmo(object_wmo* ob){ on = ob; }
    char*      str;
    int        at;
    int        weight;
    object_wmo* on;
    virtual void      at_a(){ }
    virtual void      weight_a(){ }
    virtual void      on_a(){ }
};

wmo nothing : public object_wmo
{public: nothing_wmo(object_wmo* ob) : (ob)
        { str = "nothing object"; }
}(&nothing);

wmo bananas : public object_wmo
{public: bananas_wmo(object_wmo* ob) : (ob)
        { str = "bananas"; };
}(&nothing);

wmo ladder : public object_wmo
{public: ladder_wmo(object_wmo* ob) : (ob)
        { str = "ladder"; }
        void      at_a();
}(&nothing);

wmo couch : public object_wmo
{public: couch_wmo(object_wmo* ob) : (ob)
        { str = "couch"; }
}(&nothing);

wmo floor : public object_wmo
{public: floor_wmo(object_wmo* ob) : (ob)
        { str = "floor"; }
}(&nothing);

wmo ceiling : public object_wmo
{public: ceiling_wmo(object_wmo* ob) : (ob)
        { str = "ceiling"; }
}

```

```

}(&nothing);

wmo monkey
{public:
    int      @at;
    object_wmo* @holds;
    object_wmo* @on;
};

// ***** the goals
wmo goal
{public:
    goal_wmo(){ } // stack constructor
    goal_wmo(object_wmo* ob){ object = ob; }

    int      active; // members
    object_wmo* object;
    int      to;

    // base class functions
    void      stack(object_wmo*, int, goal_wmo*);
    void      unstack(goal_wmo*);

    virtual void      active_a(){ } // attached functions
    virtual void      object_a(){ }
    virtual void      to_a(){ }
};

// ***** goal_wmo::stack
void      goal_wmo::stack(object_wmo *obj,
                          int place, goal_wmo* st_p)
{
    if(active) // copy current goal onto stack
    {
        st_p[active - 1].object = object;
    }
    active++; // store new goal
    object = obj;
    to = place;
    object_a(); // call attached functions
    to_a();
    active_a();
}

// ***** goal_wmo::unstack
void      goal_wmo::unstack(goal_wmo* st_p)
{
    if(--active) // restore old goal
    {
        printf(" restore stacked goal\n");
        object = st_p[active - 1].object;
        active_a();
        object_a(); // call attached functions
        return;
    }
    active_a();
}

```

```

}

// ***** derived goals
wmo goal_holds : public goal_wmo // ***** the holds goal
{public:
    goal_holds_wmo():(){ }
    goal_holds_wmo(object_wmo* ob) : (ob){ }
    void activate(object_wmo*);
    void satisfied();
    void active_a();
    void object_a();
}(&nothing);
goal_holds_wmo goal_holds_stack[10];

void goal_holds_wmo::activate(object_wmo* obj)
{
    stack(obj, 0, goal_holds_stack); }

void goal_holds_wmo::satisfied()
{
    unstack(goal_holds_stack); }

wmo goal_walk_to : public goal_wmo // ***** the walk_to goal
{public:
    goal_walk_to_wmo():(){ }
    goal_walk_to_wmo(object_wmo* ob) : (ob){ }
    void activate(object_wmo*, int);
    void satisfied();
    void active_a();
    void object_a();
    void to_a();
}(&nothing);
goal_walk_to_wmo goal_walk_to_stack[10];

void goal_walk_to_wmo::activate(object_wmo* obj, int place)
{
    stack(obj, place, goal_walk_to_stack); }

void goal_walk_to_wmo::satisfied()
{
    unstack(goal_walk_to_stack); }

wmo goal_on : public goal_wmo // ***** the on goal
{public:
    goal_on_wmo():(){ }
    goal_on_wmo(object_wmo* ob) : (ob){ }
    void activate(object_wmo*);
    void satisfied();
    void active_a();
    void object_a();
}(&nothing);
goal_on_wmo goal_on_stack[10];

void goal_on_wmo::activate(object_wmo* obj)
{
    stack(obj, 0, goal_on_stack); }

void goal_on_wmo::satisfied()
{
    unstack(goal_on_stack); }

```

```

wmo goal_move : public goal_wmo          // ***** the move goal
{public:
    goal_move_wmo():(){ }
    goal_move_wmo(object_wmo* ob) : (ob){ }
    void activate(object_wmo*, int);
    void satisfied();
    void active_a();
    void object_a();
    void to_a();
}(&nothing);
goal_move_wmo goal_move_stack[10];

void goal_move_wmo::activate(object_wmo* obj, int place)
{ stack(obj, place, goal_move_stack); }

void goal_move_wmo::satisfied()
{ unstack(goal_move_stack); }

// ***** the rules
// if desired object on ceiling then set up goal
// to - move ladder to object position.
rule mb1
{ (goal_holds.@active);
  ((goal_holds.@object)->on == &ceiling);
-->
  printf("mb1 goal_move object %s to %d\n",
         ladder.str, (goal_holds.object)->at);
  goal_move.activate(&ladder, (goal_holds.object)->at);
}

// if required to hold object on ceiling
// and object and ladder at same position then
// set up goal - get on the ladder
rule mb2
{ (goal_holds.@active);
  ((goal_holds.@object)->on == &ceiling);
  ((goal_holds.@object)->at == ladder.@at);
-->
  printf("mb2 goal_on object %s\n", ladder.str);
  goal_on.activate(&ladder);
}

rule mb3
{ (goal_holds.@active);
  ((goal_holds.@object)->on == &ceiling);
  ((goal_holds.@object)->at == ladder.@at);
  (monkey.@on == &ladder);
-->
  printf("mb3 goal_holds nothing\n");
  goal_holds.activate(&nothing);
}

rule mb4
{ (goal_holds.@active);

```

```

        ((goal_holds.@object)->on == &ceiling);
        ((goal_holds.@object)->at == ladder.@at);
        (monkey.@on == &ladder && monkey.@holds == &nothing);
-->

        printf("mb4 grab %s\n", (goal_holds.object)->str);
        monkey.@holds = goal_holds.object;
        goal_holds.satisfied();
    }

// if desired object on floor then set up goal - walk to object
rule mb5
{
    (goal_holds.@active);
    ((goal_holds.@object)->on == &floor);
-->

    printf("mb5 goal_walk_to %s\n",
           (goal_holds.object)->str);
    goal_walk_to.activate(goal_holds.object, 0);
}

rule mb6
{
    (goal_holds.@active);
    (goal_holds.@object != monkey.@holds &&
     monkey.@holds != &nothing);
    ((goal_holds.@object)->at == monkey.@at);
-->

    printf("mb6 goal_holds nothing\n");
    goal_holds.activate(&nothing);
}

// If you want to hold object and object on floor and
// monkey is at same position then grab it
rule mb7
{
    (goal_holds.@active);
    ((goal_holds.@object)->on == &floor);
    ((goal_holds.@object)->at == monkey.@at &&
     monkey.@holds == &nothing);
-->

    printf("mb7 grab %s\n", (goal_holds.object)->str);
    monkey.@holds = goal_holds.object;
    goal_holds.satisfied();
}

// If object is to be moved and monkey not already
// holding it, then - goal hold object
rule mb8
{
    (goal_move.@active);
    (monkey.@holds != goal_move.@object);
    ((goal_move.@object)->weight == light_c);
    ((goal_move.@object)->at != goal_move.@to);
-->

    printf("mb8 goal_holds object %s\n",
           (goal_move.object)->str);
    goal_holds.activate(goal_move.object);
}

```



```

// If object to be moved is light and object is not
// at desired place and monkey is holding the object
// then - goal walk to desired place
rule mb9
{
    (goal_move.@active);
    ((goal_move.@object)->weight == light_c &&
     (goal_move.@object)->at != goal_move.@to);
    (monkey.@holds == goal_move.@object);
-->
    printf("mb9 goal_walk_to %d\n", goal_move.to);
    goal_walk_to.activate(goal_move.object, goal_move.to);
}

// If the moved object has arrived at the desired place then
// the move goal is satisfied
rule mb10
{
    (goal_move.@active);
    ((goal_move.@object)->at == goal_move.@to);
-->
    printf("mb10 goal_move satisfied\n");
    goal_move.satisfied();
    (monkey.@holds)->at = goal_move.to;
}

rule mb11
{
    (goal_walk_to.@active);
-->
    printf("mb11 goal_on floor\n");
    goal_on.activate(&floor);
}

rule mb12
{
    (goal_walk_to.@active);
    (monkey.@on == &floor &&
     monkey.@at != (goal_walk_to.@object)->at &&
     monkey.@holds == &nothing);
-->
    printf("mb12 walk to %s\n",
           (goal_walk_to.object)->str);
    monkey.@at = (goal_walk_to.object)->at;
    goal_walk_to.satisfied();
}

rule mb13
{
    (goal_walk_to.@active);
    (monkey.@on == &floor &&
     monkey.@at != goal_walk_to.@to &&
     monkey.@holds != &nothing);
-->
    printf("mb13 walk with %s to %d\n",
           (goal_walk_to.object)->str, goal_walk_to.to);
    monkey.@at = goal_walk_to.to;
    (goal_walk_to.@object)->at = goal_walk_to.to;
    goal_move.@object = goal_walk_to.object;
}

```

```

        goal_walk_to.satisfied();
    }

    rule mb14
    {
        (goal_on.@active);
        (goal_on.@object == &floor);
        (monkey.@on != &floor);
    -->
        printf("mb14 jump onto the floor\n");
        monkey.@on = &floor;
        goal_on.satisfied();
    }

    rule mb15
    {
        (goal_holds.@active);
        ((goal_holds.@object)->on == monkey.@on);
        ((goal_holds.@object)->at == monkey.@at);
        (monkey.@holds == &nothing);
    -->
        printf("mb15 grab %s\n", (goal_holds.object)->str);
        monkey.@holds = goal_holds.@object;
        goal_holds.satisfied();
    }

    rule mb16
    {
        (goal_on.@active);
        (monkey.@at == (goal_on.@object)->at);
    -->
        printf("mb11 goal_holds nothing\n");
        goal_holds.activate(&nothing);
    }

    rule mb17
    {
        (goal_on.@active);
        (monkey.@at == (goal_on.@object)->at &&
         monkey.@holds == &nothing);
    -->
        printf("mb17 climb onto %s\n", (goal_on.object)->str);
        monkey.@on = goal_on.object;
        goal_on.satisfied();
    }

    rule mb18
    {
        (goal_holds.@active &&
         goal_holds.@object == &nothing);
        (monkey.@holds != &nothing);
    -->
        printf("mb18 drop %s\n", (monkey.holds)->str);
        monkey.@holds = &nothing;
        goal_holds.satisfied();
    }

```

What NeWS?
— or —
What light through yonder window breaks?

William Roberts
qmc-cs/liam, liam@cs.qmc.ac.uk



Department of Computer Science
Queen Mary College
London, England

William Roberts has been programming computers since age 11. After graduating from Oxford with an Honours degree in Mathematics, he worked for two years with microcomputers of various sorts, before returning to higher education. In 1985 he was awarded an MSc with Distinction by the Department of Computer Science at Queen Mary College, London, and has remained there ever since. He is currently a systems programmer supporting the Department network of over 80 UNIX machines, and actively working on NFS and NeWS.

The Network Extensible Window System (NeWS) from Sun Microsystems is a marriage between the flexibility of the PostScript page description language and the interactive capability of a modern graphics workstation. This paper describes both PostScript and NeWS, explaining how they fit together and what the implications are for writing applications under NeWS.

1. Introduction

The development of PostScript and NeWS is presented here as a derivation from some initial design goals: it isn't a historical account. Like all successful software, the development of these systems undoubtedly involved thinking about all of the stages at once and reworking ideas as work progressed.

2. PostScript

PostScript is a page description language used by a variety of different laser printers. When it first appeared, as the control language for the Apple LaserWriter, it was a radical departure from previous graphics languages and printer languages, with a new approach both to the way graphical images are formed and the work a printer should do.

2.1 The PostScript Graphics Model

PostScript was designed to work with devices that use a *raster*, a rectangular array of pixels each with a separate colour (in the case of laser printers, just black or white). Its main design goal was to provide device-independent graphics; in particular, exactly the same PostScript file should work on a device with 300 pixels per inch and a device with 2540 pixels per inch, producing the same size output though hopefully to a higher quality.

To achieve device independence, PostScript took the bold step of abandoning pixels altogether. Freed of this restraint, the authors of PostScript designed an abstract "ideal printer" and produced software that would provide the best approximation that the actual printer can manage.

2.2 The PostScript Abstraction

The idealised PostScript printer has arbitrarily high spatial and colour resolution; it can print anything as small or as large as you like, with no loss of detail, and you can have any and every colour imaginable. The printing surface is arbitrarily large, and all marks on the printing surface are made by pouring *paint* (the colour of your choice) through a *stencil* (any arbitrary shape). New marks overwrite whatever was there before and there is no way to remove a mark once you have made it.

In order to drive such a spectacular device, the PostScript language has a number of unusual features.

- The drawing surface is two-dimensional, so all positions are specified by a pair of coordinates; to allow for the limitless drawing surface and level of detail, they are always floating point values.
- The coordinates are interpreted as a position on the paper by a coordinate transformation matrix. This matrix can be changed by applying arbitrary *linear transformations*, allowing the coordinate axes to be scaled independently, made non-perpendicular etc.
- Shapes are defined by creating a *path* on the drawing surface, made out of straight lines and Bezier curves. The coordinate transformation matrix can be changed at any point as the path is made up and the path itself leaves no mark on the paper.
- The current path can be converted into marks on the paper by converting it to a stencil and pouring colour through it. The conversions possible are *filling*, which uses the current path as the boundary, and *stroking* which constructs a stencil corresponding to a line of a given width, centred along the path. When stroking the path, the ends and corners can be handled in a number of different ways to make the sections join up properly.
- To make the transformation matrix easy to modify and easy to put right again, the complete *graphics state*, the current colour, the current path, the prevailing coordinate transformation etc., can be pushed onto a stack and later restored when necessary, eliminating any changes made in the interim. The results of any stroking or filling remain on the page and cannot be undone.

The stencil and paint model of graphics makes it easy to produce very efficient fonts for printing text: each character is described in terms of PostScript paths at some generic size, and the coordinate transformation mechanism is used to transform those paths into any desired size and orientation.

In addition to all the above "necessary" features, the designers of PostScript added an number of "extras":

- PostScript is a full programming language complete with iteration, recursion, dynamic binding and procedural abstraction. Unlike earlier printer languages where single groups of input characters produced a single effect on the output, an element of a PostScript program can be executed many times before the output is complete. This makes simple things harder, for example you need to write a PostScript program to print "plain text" rather than just sending the plain text itself, but it makes hard things much easier: to convert from some other graphics language into PostScript is normally a matter of writing PostScript routines to simulate the primitives of the other language, and then perform very simple substitutions on the other language to call the appropriate PostScript routines.
- PostScript is an interpreted language capable of self-modifying programs. It uses dynamic binding to find the meaning of names at execute time, using a stack of *dictionaries* which contain (*name, value*) pairs. To resolve a name, PostScript looks in the top dictionary on the stack, then the next and so on down until the name is found. This mechanism is so flexible that it is possible to implement object-oriented programming constructs such as classes and instances, complete with inheritance and message passing, all in standard PostScript.

2.3 PostScript under the Hood

The idealised PostScript printer is impossible to build, so all PostScript implementations are just approximations. How does a 300 dots per inch black-and-white raster printer produce an acceptable implementation of the ideal?

- Laser printers essentially transfer a prepared bitmap into a printing engine of some kind, so the PostScript interpreter has a *framebuffer* bitmap which represents the finished page. The default transformation matrix converts from the default PostScript coordinates (1 unit = 1/72nd of an inch, y increases up the page and the origin is in the bottom left hand corner) to the framebuffer pixel coordinates. The PostScript transformations build on the existing matrix, rather than requiring the programmer to supply a replacement matrix, so PostScript programs do not need to look at the default matrix. When a stroke or fill primitive is executed, the transformation system produces directly the framebuffer coordinates needed to change the appropriate bits in the bitmap.
- Curved portions of paths are approximated by straight lines, subject to a *flatness* threshold which limits the maximum discrepancy between the ideal curve and the straight line portion.
- Colours are represented on a monochrome system as intensities, rather like watching colour TV pictures on a black-and-white TV set. In order to produce shades of grey on a device where pixels are either black or white, a printing technique called *halftoning* is used: the page is divided up into spots, and each spot has a different size or shape according to the prevailing gray level. This is the same way the photographs are done in newspapers and relies on having a reasonable number of printer pixels per spot. On the Apple Laserwriter the spot is large enough to distinguish 64 different gray levels, and the intensity is approximated to the nearest level.
- The stencil-and-paint model is clearly quite expensive in computational terms, and would be very inefficient for printing pages of text (such as the one you are reading). To optimise this particular use of PostScript printers, the PostScript interpreter maintains a *cache* of recently used character shapes and can simply reuse a *scan-converted* character rather than recompute the bitmap when all that has happened in a simple translation. This increases the speed of printing character shapes by a factor of 1000 or more.

All of these approximations are kept hidden from the PostScript programs: a determined and nosey program can deliberately discover some of this information, but most PostScript programs operate under the assumption that their initial environment is correct and that they can just get on with their printing. This makes it fairly easy to write PostScript code that changes the initial environment and so causes subsequent code to produce a modified output: a useful example is changing things so that text formatted for an A4 page comes out reduced and rotated as two A5-sized prints per piece of A4 paper.

There is yet more to PostScript, but the above description covers most of the salient points. It has been endorsed as a standard printer page description language by such diverse companies as Apple, IBM and the phototypesetter manufacturers Linotype. The range of PostScript printers available spans from cheap(ish) 300 dot per inch printers up to the Linotronic 300 which can print on A4 width continuous rolls of paper or film, at a resolution of 2540 dots per inch. It is almost universal practice to use a 300 dot PostScript printer as an accurate page proofing device prior to printing on the expensive Linotronic; exactly the same PostScript file is used in both cases.

3. NeWS — PostScript on the Screen

Sun Microsystems have produced a PostScript interpreter which uses the Sun bitmapped screen as its drawing surface: all of the PostScript approximation techniques used for the laser printers apply just as well to a screen raster at a resolution of only 80 dots per inch. Not content with simply using the Sun screen as a way of previewing printer output, Sun have extended the PostScript language so that it can be used to provide all the features of a conventional window manager, and produced a *graphics server*, a process which deals with all the physical screen and input device handling, presenting a device independent interface to application programs. This server is called NeWS and can be used in place of the older Suntools window manager.

3.1 What did Sun add to PostScript?

A modern bitmapped display is normally provided with a number of libraries which provide basic graphics functions such as drawing lines and filling areas, the ability to define independent overlapping rectangular areas on the screen, on which applications can do their graphical stuff without worrying about who else is drawing what where, and lastly facilities for interactive graphics, such as notifying an application program that the mouse pointer has been moved into its window. PostScript has all the drawing primitives, but none of the others.

- NeWS generalises PostScript by providing multiple drawing surfaces rather than just the one. These *canvas* objects are arranged in a tree hierarchy with the screen framebuffer at the root. There are primitives to create a new canvas, complete with its default transformation matrix and an arbitrary shape. Each canvas has a *parent* canvas and will appear on the screen on top of its ancestors, blocking out any part of an ancestor that is behind it. The ancestors get their own back, however, by restricting the portions of their descendants that are actually visible to those pixels that are common to all ancestors (in graphics parlance, the child canvas is *clipped* to its parent when the screen image is composed). Canvasses are very efficient and can be arbitrary shapes, including disconnected regions; Sun are fond of using their logo as a canvas shape.
- The output of a PostScript printer is a completed page, so it makes no sense to work on separate parts simultaneously. On the screen however, it is often desirable to see things in progress (for example program listing scrolling in a

terminal window or some pretty piece of animation), so NeWS adds *lightweight processes* to the PostScript model. The process primitives are not unlike UNIX; a new process is created by specifying a procedure for the new procedure to execute and applying the `fork` primitive (corresponding to a combined `fork()` ... `exec()` sequent of UNIX system calls). The result of a `fork` is the creation of the new process, with a process object returned to the parent process. To collect a return value from the child, the parent can execute the `awaitprocess` primitive. The PostScript dictionary structure provides an extremely flexible form of shared memory; two processes can share any object for which they both have a name. Private memory is produced by creating a new dictionary and not telling anyone else about it!

- Interactive systems by definition require input. In NeWS, this is provided by a mechanism of *events* in which each input such as a key being pressed or the mouse moving is converted into an event object and distributed to any process that is interested. Processes can declare an interest in various classes of event and can generate their own new types of event for special purpose. This mechanism is sufficiently flexible to allow all sort of uses: timers are implemented by creating events with a "do not deliver before" timestamp that is some time in the future.

In addition to these extensions to the language, NeWS is supplied with a selection of library routines written in NeWS PostScript, covering traditional library facilities such as pop-up menus, rectangular windows with frames, terminal emulation, and *toolkit* items such as buttons, text input items, sliders etc.

The above description of NeWS is intended to give a general feel for what it involves and skips many details: a detailed Technical Overview is available from Sun and the interested reader could also look at the "comp.windows.news" USENET bulletin board. By way of rounding of this section, here is a recent piece of work of which the author is rather proud:

Using the mouse, point to a position on the screen. A white circle appears, followed by a smaller black circle on top. These are both canvasses, and the black circle is a child of the white one. Next, a PostScript process is created which expresses an interest in any movement of the mouse. When the mouse moves, this process is notified and tries to move the black circle to the current cursor position, subject to the constraint that the centre of the black circle must never go outside the white circle. Recall that the child canvas is clipped to the intersection of its ancestors, so the bits of the black circle that fall outside of the white circle are not displayed on the screen. The net result? An "eyeball" that watches the cursor as it moves around the screen, all done in PostScript and not a UNIX program in sight other than the NeWS server itself*.

4. *Programming Applications to run under NeWS*

From one viewpoint, writing an application to run under NeWS is very like writing one to run under any other system: you produce interface routines between your application code and the graphics system, each of which produces sequences of graphics primitives corresponding to the objects manipulated by the application. For fancy interactive graphics, your application program will probably take essentially

* I leave to the reader's imagination what I then did to the Mona Lisa demonstration....

the form of a central event handling loop which calls appropriate routines in response to combinations of inputs.

All of this is possible under NeWS and just as simple as with any other low-level "draw a line from x1,y1 to x2,y2" system. However to use NeWS in this way is an awful waste of a good system.

The NeWS server is **programmable**: you can send programs to it that work in parallel with your application code. The first way in which this can be used is to push the graphics interface routines into the NeWS server itself. By sending a PostScript procedure definition to the server, the application can add "customised graphics primitives" to the repertoire of the server; if your application draws lots of symbols on the screen in various places, then the PostScript routines might be "draw pine-tree-symbol at (x,y)" and so on.

The second way PostScript programs can help an application is by putting the fast interactive bits of the application entirely into the server. A classic user-interface technique is called *rubber-banding*: if a rectangular area is required, let the user define it by choosing opposite corners. After the first corner has been selected, draw on the screen the rectangle that would result if the current cursor position was used as the other corner. Because the rectangle chosen changes as the mouse moves, it looks as though the boundary is a stretchable line, hence the name rubber-band line. Using a conventional system, every time the mouse moves, the previous rectangle has to be removed and the new one drawn. By writing a PostScript procedure to do just this rubber-band rectangle task, the application is freed from having to interpret the events and do all the drawing and undrawing. Once the user has chosen the two corners, the PostScript procedure sends back the 4 coordinates and the application can carry on.

5. Parting Thought...*

The connection between the NeWS server and application programs is by a simple byte stream, such as a network connection or a serial line. Careful use of PostScript programs in the server means that only small quantities of data are sent through that connection. If someone ports NeWS to a cheap computer with good graphics, we could all run NeWS in the "terminals" on our desks but attached to the dumb terminal computers we already have.

What price a graphics console at home talking through a phone line to a number-cruncher at work?

* It would be too confusing to say "Postscript" in this context!

The X/OPEN show in Luxembourg revisited

Martijn de Lange
martijn@ace.nl

ACE Associated Computer Experts bv
Amsterdam, The Netherlands

Having read Theo de Ridder's report on the X/OPEN demonstration event in Luxembourg (EUUGN, Vol7 No1) experienced readers may be suspicious as to what was actually demonstrated in Luxembourg.

It is with the permission of the X/OPEN group that I summarise some of the background to the demo here.

The purpose of the event was to show the installation, compilation, and execution of an application on all 11 members' systems without system specific adaptations.

The 20/20 spreadsheet from Access Technology was selected as the application for the demo because it is well known and because a spreadsheet provides a visible and lively demonstration. As was emphasised during the question and answer session on the first demo-day in Luxembourg, and as was confirmed by Paul Basson of Access Technology, every installation to date of 20/20 on a new machine had been done as a separate specific port. Each port required adaptation of the sources. This is exactly what X/OPEN is aiming to avoid.

The systems as used during the demo were all different: in hardware as well as in their UNIX implementation. All X/OPEN members are in the process of modifying their UNIX implementations in order to conform to the X/OPEN standard. Their joint commitment is to complete this by the third quarter of 1987.

ACE was given the task to create one set of sources that would compile and run identically on all 11 systems. For this reason the 20/20 sources were modified so that they would generate a 20/20 executable on each system: using the same includes, the same makefile, and the same C sources.

The preparation of the demo and the exercise of modifying and testing took a considerable amount of time. It was done using special version-control tools on top of `sccs`, specially developed "define-strip" tools, a special devised version of `cpp`, and our makefile generator `gendep`. `uucp` was used as the means for communication between the 11 systems. The reason why none of these porting and control tools were shown is simple: it was not the purpose of the demonstration. The purpose was "one set of sources, make, and run".

Much system- or machine-dependent code was re-done in a portable fashion, and care was taken to utilise only those facilities common to all 11 UNIX implementations: a subset of the X/OPEN standard.

The established changes were coded in under:

```
#ifdef XOPEN
```

```
#endif
```

a define that was then given in the (one) makefile and that was cleverly noted by Theo.

We considered it right to make the X/OPEN dependent changes visible. Theo noted the `-DXOPEN` in all compilations and perhaps he noted a couple of other defines as well; however, they were all identical on all systems. We expressly displayed the make-trace to show that exactly the same compile steps were performed using the same sources and the same defines. It is very difficult to execute different compilations by just typing "make" when the makefile and sources are identical, which I can assure you that they were.

We could also have flagged our changes by

```
#ifndef NOT_XOPEN
```

```
#endif
```

and then not defined `NOT_XOPEN` in the makefile. Would this not trigger suspicion?

As an introduction to the installation, the sources were read into some systems from floppy, just to demonstrate that X/OPEN bothers about source-code transfer. This may not seem important, but ACE having, much experience in the porting of software, knows that very often floppies (and other media) cannot be read on different UNIX systems. In fact source transfer has always been a major problem area when porting software to different systems.

We considered having 11 mag-tape units on stage but decided that it would look like a bad late-sixties science fiction movie. We then considered using an (Ether)network, but that was not (yet) within the X/OPEN standard, so we got stuck with the floppies as the only feasible exchange media to demonstrate. It is a sad world indeed. But the fact that X/OPEN defines formats and standards for media is a small step in the right direction anyway.

In Luxembourg the audience witnessed the effort of 11 computer manufacturers (eleven: Theo's list plus DEC) whose systems are not yet all X/OPEN conformant, demonstrating that one could achieve true portability over all systems for a medium size software product, provided the product is written according to the X/OPEN standard.

The fact that considerable effort was required to create such a portable version of the demonstrated application does not affect the X/OPEN message. It merely stresses the urgent needs: on one side, for defined and standardised interfaces, and on the other, a much better appreciation by software engineers of what portable software is.

Without these two going side-by-side, the way to go will remain long forever.

List of Interesting UNIX Publications and How to Obtain Them

Wytze van der Raay (NLUUG)
wytze@gouldnl

Gould European UNIX Support Centre
Computer Weg 4
3606 AT Maarssenbroek
The Netherlands

Wytze van der Raay is the current treasurer of the NLUUG. He is also in the process of starting up a European UNIX Support Centre for Gould. He has been active in the UNIX arena since 1980, working his way up from version 6, in both academic and commercial environments.

Here is a list of some interesting publications about UNIX, along with details on how to get them. Please send any updates or suggestions to myself at the above address.

Article Collections

1. *Bell Labs Technical Journal 1978 UNIX Issue*

Price: 23 US \$
Via: UNIX Europe Ltd.
27A Carlton Drive
London SW15 2BS
England
+ 44 (1) 7856972

2. *Bell Labs Technical Journal 1984 UNIX Issue*

Price: 23 US \$
Via: UNIX Europe Ltd.
27A Carlton Drive
London SW15 2BS
England
+ 44 (1) 7856972

3. *Proceedings of USENIX Conferences*

a.	Phoenix,	Summer 1987	20 US \$	1	25 US \$	shipment
b.	Washington,	Winter 1987	20 US \$	1	25 US \$	shipment
c.	Graphics,	December 1986	10 US \$	1	15 US \$	shipment
d.	Atlanta,	Summer 1986	10 US \$	1	25 US \$	shipment
e.	Denver,	Winter 1986	10 US \$	1	25 US \$	shipment
f.	Graphics,	December 1985	3 US \$	1	7 US \$	shipment
g.	Dallas,	Winter 1985	10 US \$	1	25 US \$	shipment
h.	Graphics,	December 1984	3 US \$	1	7 US \$	shipment
i.	Salt Lake City,	Summer 1984	10 US \$	1	25 US \$	shipment

Via: The Secretary, EUUG
Owles Hall
Buntingford, Herts
England SG9 9PL
+ 44 (763) 73039

4. UNIX Papers (edited by The Waite Group)

ISBN: 0 672 22578 6
Price: ~ Dfl. 75,— (US: \$26.95)
Via: Your local bookshop, or:
Howard W. Sams & Company
A Division of MacMillan Inc.
4300 West 62nd Street
Indianapolis, Indiana, 46268 USA
+ 1 (317) 298 5699

Standards

1. *System V Interface Definition (SVID), Issue 2*

Volume 1, code 320-011
Volume 2, code 320-012
Price: 245 Dfl. excl. VAT
Via: Your local Olivetti representative, or:
Olivetti Nederland B.V.
OEM/VAR Division
Olivetti — AT&T Products
Verbeekstraat 19-21
2332 CA Leiden
The Netherlands
+ 31 (71) 319668

2. *X/OPEN Portability Guide*

Second Revised and Updated Edition 1987 (in 5 volumes)

ISBN: 0 444 70179 6
 Price: 350 Dfl excl. VAT
 265 Dfl excl. VAT via special NLUUG conference form
 Via: Your local bookshop, or directly from the publisher:
 North-Holland
 Attn: J. Dirkmaat
 P.O. Box 1991
 1000 BZ Amsterdam
 The Netherlands

3. *IEEE 1003.1 UNIX Trial Use Standard*

Price: 19.95 \$ + shipment (stock number SH10546)
 Via: IEEE Service Center
 445 Hoes Ln.
 Piscataway, NJ 08854
 + 1 (714) 821 8380

4. *1984 /usr/group Standard (out of date)*

Price: 15 \$ + 20 \$ shipment
 Via: /usr/group
 4655 Old Ironsides Drive, Suite 200
 Santa Clara, CA 95054
 USA
 (Money Orders, Visa & Mastercard payments are accepted)
 + 1 (408) 986 8840

System Documentation

1. *AT&T System V Release 2.0 Documentation*

	Title	Price	Order code
a.	Basic Documentation Set	249.95 US \$	307-080
b.	Supplemental Documentation Set	140.00 US \$	307-082
c.	VAX Processors Version 2.0 Supplement	25.00 US \$	307-220

Via: UNIX Europe Ltd.
 27A Carlton Drive
 London SW15 2BS
 England
 + 44 (1) 7856972
 Or: AT&T Customer Information Center
 Commercial Sales Representative
 P.O. Box 19901
 Indianapolis, Indiana 46219

+ 1 (317) 352 8556

2. *Berkeley 4.3 Documentation*

- | | | |
|----|---|---------------------|
| a. | 4.3 User's Manual Set (3 volumes) | 25 US \$ per volume |
| b. | 4.3 Programmer's Manual Set (3 volumes) | 25 US \$ per volume |
| c. | 4.3 System Manager's Manual (1 volume) | 10 US \$ per volume |

Via: Howard Press
 c/o USENIX Association
 P.O. Box 2299
 Berkeley, CA 94710
 USA

- Prices are excluding shipment (very expensive!).
- Only for USENIX members.
- Order form in "login:" (see "Magazines"). When ordering please specify your USENIX number. The form should be signed by an "Authorised Institutional Representative", and include your address and bill address.
- USENIX office phone: +1 415 528 8649
 email: {ucbvax!decvax}!usenix!office

3. *Nutshell Handbooks*

- | | | |
|----|--|-------------|
| a. | Managing UUCP and USENET (Grace Todino & Tim O'Reilly) | 18.00 US \$ |
| b. | Using UUCP and USENET (Grace Todino) | 15.00 US \$ |
| c. | Reading and Writing Termcap Entries (John Strang) | 9.00 US \$ |
| d. | Programming with Curses (John Strang) | 9.00 US \$ |
| e. | Learning the Vi editor (Linda Lamb) | 15.00 US \$ |
| f. | Managing Programs with Make (Steve Talbott) | 9.00 US \$ |
| g. | UNIX in a Nutshell, A Desktop Quick Reference for Berkeley | 19.50 US \$ |
| h. | UNIX in a Nutshell, A Desktop Quick Reference for System V | 19.50 US \$ |

Shipment: 2.50 US \$ (0-4 books)
 3.50 US \$ (5-10 books)
 4.50 US \$ (11-25 books)

Via: Nutshell Handbooks
 O'Reilly & Associates, Inc.
 981 Chestnut Street
 Newton, Massachusetts 02164
 + 1 (617) 527 4210

Product Catalogues

1. *UNIX Europe Ltd Software Catalogue*

Price: 26 US \$
Via: UNIX Europe Ltd.
27A Carlton Drive
London SW15 2BS
England
+ 44 (1) 7856972

2. *UNIX System V Software Catalogue (US)*

Price: 26 US \$
Via: UNIX Europe Ltd.
27A Carlton Drive
London SW15 2BS
England
+ 44 (1) 7856972

3. *UNIX products for Europe*

ISBN: 90 14 03522 5
Price: ~85 Dfl
Publisher: Samsom
Postbus 4
2400 MA Alphen a/d Rijn
Via: Your local bookshop, or from the EUUG:
Send cheque for 15 UK £ to:
The Secretary, EUUG
Owles Hall
Buntingford, Herts SG9 9PL
England
+ 44 (763) 73039

4. *EUUG Micros Catalogue Edition 2 1985*

Price: 7.50 UK £
Via: EUUG
Send cheque for 7.50 UK £ to:
The Secretary, EUUG
Owles Hall
Buntingford, Herts
England SG9 9PL
+ 44 (763) 73039

5. *UNIX Products Directory Winter 1987*
(*/usr/group's Sixth Edition*)

Price: 50 US \$ + 20 US \$ shipment
(Members of /usr/group: free, or 25 US \$ + shipment)
Via: /usr/group
4655 Old Ironsides Drive, Suite 200
Santa Clara, CA 95054
USA
(Money Orders, Visa & Mastercard payments are accepted)
+ 1 (408) 986 8840

Magazines

1. *Unigram/X, the weekly newsletter for UNIX system users*

Price: 225 UK £ per year (EUUG member price)
300 UK £ per year (others)
Via: Freepost 32
Unigram Products Ltd.
London W1E 6Q2
England
+ 44 (1) 439 1632

NOTE: The member price is only applicable if you state that you are a member of the EUUG.

2. *UNIX/World*

Monthly.
Price: 18 US \$ + 20 US \$ shipment
Via: UNIX/World
Subscription Services
P.O. Box 1929
Marion, OH 43306
USA
+ 1 (415) 940 1500

3. *UNIX Review*

Price: 55 US \$ (95 US \$ for airmail)
Via: UNIX Review
Miller Freeman Publications Co.
500 Howard Street
San Francisco, CA 94105
USA
+ 1 (415) 397 1881

4. ;login:

Two-monthly magazine of the USENIX Association
Price: 16 US \$ per year
Via: The Secretary, EUUG
Owles Hall
Buntingford, Herts
England SG9 9PL
+ 44 (763) 73039

5. *IX Magazine*

Price: 25 UK Pounds (free within the UK)
Via: IX magazine
BTB Mailing Services Ltd.,
Unit 15, The Manton Centre,
Manton Lane,
Bedford MK41 7LX
England

6. *UNIX Info*

(In Dutch language)
Price: Free (controlled circulation)
Via: Sala Communications
Den Texstraat 5a
1017 XW Amsterdam
+ 31 (20) 273198 or + 31 (2152) 63431

Tapes

Please see Frank Kuiper's article elsewhere in this issue.

Via: EUUG Distributions
c/o Frank Kuiper
Centrum voor Wiskunde en Informatica
Kruislaan 413
1098 SJ Amsterdam
The Netherlands
+ 31 (20) 5924056 or + 31 (20) 5929333
Email: euug-tapes@mcvax or frankk@cw.nl or ...!mcvax!frankk
(see also: this EUUG Newsletter)

Hydrological Equilibrium (The EUUG Conference at Dublin)

Joe Nicholson
joe@inset.co.uk



The Instruction Set
London, England

Joe Nicholson is a technical consultant with The Instruction Set involved in operating systems, networking and relational databases.

He originally studied Computer Science at University College London and has never quite recovered.

Coming from an Irish background, he is well versed in Irish culture and drinking habits. In September he spent five days sampling the black nectar and found time to attend the EUUG conference in Dublin.

Sunday

I arrived in Dublin about six p.m., with a colleague from Inset, Harnish Patel, flushed with anticipation for my very first UNIX conference. Being a virgin, so to speak, I didn't quite know what to expect apart from sandals. After a rather arduous flight from London and a brisk taxi ride into the centre of Dublin, we were confronted by hoards of green and gold clad revelers brandishing pints of the black nectar. What were they celebrating? A new licensing agreement from AT&T, 4.4 would have the vi(2) system call, or that David Tilbrook was staying in America *and* had just bought a drink! The truth was that Meath (a county adjacent to Dublin) had just won the All Ireland Gaelic Football Final at Croke Park. The red and white people (Cork), were later seen drowning their otherwise buoyant sorrows in the bars of Dublin.

Registration for accommodation was a surprisingly harmless affair in an exceedingly quiet Trinity College. Still not a sandal in sight! The accommodation was modest for hotel standards but rather luxurious compared with most student accommodation. A quick recce around the campus still revealed no sandals, so we had a stroll around town. An excellent repast was had at the famous Scottish restaurant chain, McDonalds. A few bars later we ended up at a very crowded side street pub, crammed with Cork supporters. This was the real start of my hydrological equilibration program, although Harnish started modestly, preferring "shorts" of Bailey's Irish Cream. Only in Ireland do you get Bailey's Irish Cream served in half pint glasses!

We retired reasonably early, because that was the time the pub shut.

Monday

I started the day with a full fried Irish breakfast, with black pudding and wholemeal scones. There were three full day tutorials on Monday, starting at 9.30: Streams, 4.3 Internals and Advanced Shell Programming. I opted for the streams, given by Bob Duncanson from UEL, and Harnish went to the 4.3 internals, given by Kirk McKusick and Mike Karels from Berkeley. The streams tutorial was a good introduction to streams programming, making the (fairly reasonable) assumption that you were a system programmer. Everyone came out with the knowledge that stream buffers were allocated from a small lined memo pad and were linked together with a linked list of staples.

Discussion in the college bar later that evening revealed that most the people who attended the 4.3 internals tutorial were well pleased. Many complained, however, of a distinct lack of mbufs, although a few pints of Guinness soon compensated for the imbalance. Later that evening, everyone was feeling a bit peckish, so we split into nice small manageable groups of about 20, to go and find somewhere to eat. About 15 of us ended up at a rather posh curry house in some dodgy looking side street. A prize for the best shoes of the conference must certainly go to the doorman of the restaurant who was in traditional Indian uniform, wearing a pair of shoes which looked like winkle pickers doing the loop the loop. After an okay meal, some of us made it back to the Buttery for a last glass of milk before bed.

Tuesday

I tried to start the day with a full fried Irish breakfast, with black pudding and wholemeal scones. I got to the desk and the man said "One pound please". I said "But I had this for free yesterday". He replied "Can't you read. There is a big notice saying what you can have for a continental breakfast outside. Can you read Sir?" Well, I suppose cereal and coffee is much better for you anyway. The seminars organised for the Tuesday were of a much lower technical content than those organised for the previous day, so I decided, like many others, to soak up some of the Dublin sunshine. Sitting, reading on a sunny day in Trinity College is certainly one of the nicer sides to life.

After the conference lunch, a largish group of pilgrims gathered in preparation for the sacred pilgrimage to the Guinness brewery. Alain Williams, the EUUG newsletter editor, was one of the brave group of worshippers who formed the advanced party. He returned with the good news that the brewery was still standing and served excellent Guinness (for free !!). He was our guide and even entered the temple for a second time. What dedication.

The Guinness brewery is unlike many breweries, which when you visit them, you have to go on a long trek round, being shown large metal casks, pipes and valves. Guinness have streamlined this operation into a 5 minute film, which you don't have to watch, and then as many halves of Guinness as you can drink, before closing time at 4 p.m.

Experience from the previous night indicated that a group of 20 was rather unmanageable for dinner. I ended up with a group of about seven others at a Lebanese restaurant with excellent Lebanese cuisine served by waitresses with thick Irish accents. A spicy meal was had by all and again some of us made it back to the Buttery for a last glass of milk.

Wednesday

The conference proper. The people who registered the night before had some idea of the bombshell which has to hit the regular EUUG delegates on Wednesday morning. The comments sounded more like the press release for some new yuppy UNIX box. "Not backward compatible with previous versions", "Very portable", "Easily fits on

a shelf". All these comments were of course referring to the newly modelled, slimline cabriolet version of the EUUG conference proceedings. This means it may look a bit stupid on your shelf, but it is a lot easier to lug round the conference for three days.

The conference was introduced by the EUUG president, Teus Hagen. His reminiscences of old EUUG letters found in a shoe box ended with announcements of the new EUUG product line, modelled by internationally famous running star, Kirk McKusick. Each item had a number on it describing its street credibility. Marks, of course, out of ten.

The unnerving responsibility of giving the first talk fell on a young French student, Solange Karsenty, who opened with a talk on the lines of the conference theme — User Interfaces. Fun and games started right from the beginning... 'Can you <crackle> me', "IS THIS TOO LOUD", "Don't stand near the podium", "Stand with one hand down your trousers", "Oh sod the microphone <clunk>". Yes it was a new audience participation game — "Opportunity Microphones".

Mike O'Dell tried to get some fur flying with his talk "What they don't tell you about window systems". Most people were more shocked with the fact that he wore a tie. He did admit that he felt "a bit isolated over the water", much to everyone's amusement.

Chris Campton came up with a new acronym to use as a chat up line at UNIX parties, during his MUSK talk. WYSIWIS — What You See Is What I See.

Each session was introduced by different representatives from the User Groups which make up the EUUG. The general idea was that the session chair would tell something about their different organisations, although some did prefer to show videos of their army training! One of the session chairs came up with the reason why you *have* to drink Guinness in Ireland. As it rains so much Ireland, you have to compensate for this external hydrological excess by consuming some internal hydrological excess in the form of the black nectar. This is backed up by extensive experimental observations: when it starts to rain, everyone goes into the pub to drink Guinness.

After lunch, the ever present Dan Klein ruined many years of detente in Europe with his UBOAT (UNIX Based On-line AIDS Tutorial). Dan admitted that "it is quite tricky to get vi(1) to work from a pipe". What is he doing writing an AIDS tutorial anyway?

I then skipped a couple of talks and wished I had skipped three. Fortunately, Neil Groundwater rounded off the day with a very interesting talk on a system to annotate histology slides for hospitals, accompanied with some very gory looking colour photos. The hardware description was very interesting, especially as some devices were connected to the "sexy" bus (SCSI if you have your dentures in). Whether the system is accepted by hospitals depends on the head doctors, where "the guy with the whitest hair always wins". The top ten histology CDs should be available at your local pathology lab, soon!

Software Ireland had kindly arranged a "reception" (s/r.*n/pi..up/g) for the conference delegates in the Atrium, a wonderful wooden "theatre" attached to the side of the Buttery Bar. Refreshments (s/r.*s/booze/g) were served from 8.30, time for everyone to grab a bite to eat and have a shower. A traditional Irish folk band played ... traditional Irish folk music, to a very receptive audience in one corner of the Atrium. This was despite that fact that Dan Klein had taken his shoes off! The evening really got swinging when the band moved upstairs, where they could be heard. Some of the committee members and assorted others tested

their coordination skills at group dancing, on a rather narrow balcony. The results were somewhat less than traditional.

Thursday

Thursday kicked off with a talk by Allan Milne on BNF definitions. This in itself deserves a medal as Allan did have a drop of Guinness the night before. By all accounts, it was a interesting talk. After lunch, Roger Bivand demonstrated some of the varied requirements that geographers have for user interfaces. He sited a case where a researcher had the use of a Cray to do some analysis of the correlation between a disease and location. The program churned away for a couple of days and "the result was ... that they destroyed the printer". One requirement for geographers must therefore be stronger printers!

Mike O'Dell came back for his second talk of the conference, "What they don't tell you about Hawaiian shirts before you buy them". He described HUB, a lightweight process operating system. The sort of thing "you ought to be able to write in an afternoon". It is, however, a "Scouts' Honour programming environment", like most real time executives.

The boys from Berkeley saved the day by filling in for a non-speaker, with what they didn't have time to say at the 4.3 internals tutorial — Berkeley Futures, soon trading on Wall Street. Some interesting things included:

1. Only 1 kernel allocator (instead of 14);
2. Getting rid of some of "Kirk's armpit variables";
3. Remote filesystem layer, à la vnodes;
4. Layering of device drivers, à la streams;
5. Process debugging (/proc), à la V8, and
6. Religion server, à la Allah.

Thursday finished up with a ramble through Micheal Hawley's favourite pastime — music. Who else has a utility called ntt — Name That Tune? The main theme was reproducing pianos and with the help of a rather natty video, everyone was able to don miners' helmets and crawl under Mike's piano. He also freely advertised a 1500-pipe organ (seems like a big boy), as long as you had a cathedral to put it in.

Thursday evening was the EUUG 10th anniversary dinner at the Royal Free Hospital, in the suburbs of Dublin. Most people didn't know what to expect, but even so, when they got there it certainly wasn't what they expected. The function was held in the cellar and consisted of little stalls serving different tastes of Irish cuisine, from smoked salmon to potato stew. Yes you guessed it, Guinness was served as well. Delegates moved from stall to stall choosing what they fancied and chatting with different groups of people. Street sellers wandered around pushing carts of fruit, singing "Cockles and Mussels". Shoe shines offered to clean your shoes, and the whole atmosphere was that of Dublin of yester year.

During the proceedings, the American representative of USENIX presented the EUUG chairman, Teus Hagen, with a poster depicting the real motivation and drinking force behind the EUUG. The poster illustrated beer around the world.

The general opinion was that this type of function was much better than a full sit-down meal where you have to make conversation with the same people for 2 hours.

Friday

The last day of a very short week. Lori Grob introduced the slide command (`slide(1M)`). The command is invoked at the end of each overhead, whereupon you shout "slide" and your assistant puts on the next slide. A trend for future EUUG conferences? Perhaps there could be a call for sliders, as well as a call for papers. Lori admitted that some of the problems with automatic exploitation of concurrency "sort of reduce to the halting problem". I wonder what her next next paper will be on.

Mr Sleaze (Bart Locaithi) revealed that people have problems with his name. "Telephone operators around the world think my name is Mark McKenzie". His talk on the implementation of the routine `bitblt()` would make most C programmers choose something simple like APL. Do whiles inside switch statements, jump tables, text in the data, `cpp(1)`, `asm()`'s, and all from a guy who says he can't program in Assembler.

Mr. \zr (David Tilbrook) gave us "Cleaning up UNIX source", subtitled "Life without QED". Most of his talk was about the new release of Andrew. Why is he in captivity anyway? David dealt admirably with the loaded question from the front row. It appeared from his barrage that David doesn't like install scripts which tell you you are running on 4.2, when you are actually running MS-DOS.

The conference wound up with the extremely brief business meeting. The winners of the highly competitive EUUG quiz were announced and the winning solutions are listed below. A new `errno` was introduced by Peter Collinson: `EOLLIE` Kernel failure. The conference ended with a runner carrying the symbolic flaming EUUG travel bag leaving Dublin on route for the second half of the EUUG 10th anniversary in London.

As expected, people drifted away to catch planes home, quickly exchanging business cards they will never use. My thoughts on my first UNIX conference were clouded by the fatigue of 5 days of sitting around and socialising, but I thoughtfully enjoyed myself and met some very nice and interesting people. I just hope I make it to the EUUG in London.



Euro/American Relationships being furthered

EUUG Spring 1988 Conference "UNIX around the World"

The European UNIX systems User Group (EUUG) Spring 1988 Conference will be held in London, England from Wednesday 13th to Friday 15th, April. The venue is the prestigious Queen Elizabeth II Conference Centre situated opposite Westminster Abbey and the Houses of Parliament.

The Conference, entitled "UNIX around the World", is attracting the very highest quality technical papers thus affording an international forum for the presentation and exchange of current work on a wide variety of topics related to the UNIX system and C language. The global nature of the Conference stresses the importance of standards, portability, security and communications. Technical presentations concerning SVID, X/OPEN, Posix and ANSI C are of particular interest to prospective attendees as well as talks on Secure UNIX, UNIX Networking and real-time UNIX. Authors have been found in many different countries where UNIX licenses exist, for example: Australia; Canada; Israel; Japan and the USA; in addition to the many speakers from EUUGs National Groups.

The list of internationally known speakers, many of whom were original contributors to the UNIX programming environment, will include:

- * Peter J Weinberger (USA), Bell Research Laboratories.

Co-author and the W in the book "The AWK Programming Language". Peter is also known for the f77 I/O library and the network face server, *faced*. Peter's work in the 8th and 9th edition systems included an unbounded precision arithmetic package, *mp*, a fast factoring program, *afactor*, a B-tree library, *cbt*, and a new code generator for C. Above all, his v8 network file system bound a stable of machines together into a logically homogeneous system.

- * M Douglas McIlroy (USA), Bell Research Laboratories.

Author of the papers "The UNIX Success Story" and "A Research UNIX Reader", Doug exercised the right of a department head to muscle in on the original two-user PDP-7 system. Later he contributed an eclectic bag of utilities: *tmg* for compiler writing, *speak* for reading text aloud, *diff*, and *join*. He also collected dictionaries and made tools to use them: *look* (V7), *dict* (V8), and *spell* (V7).

- * Maurice J Bach (Israel), IBM Scientific Centre.

While with AT&T, Maurice wrote the much praised book "The Design of the UNIX Operating System". This unique, authoratative text documents the internal algorithms and structures which form the basis of the kernel, and analyses their relationship to the programmer interface. Much of the material was based upon courses Maurice taught at AT&T.

- * Michael E Lesk (USA), Bell Communications Research.

With a prescient market instinct, Michael made text formatting accessible to the masses with the generic macros `-ms`, which were to `troff` what a compiler is to assembly language. He rounded out `-ms` with the preprocessors `tbl` for typesetting tables and `refer` for bibliographies. He also made the `lex` generator for lexical analyzers. Eager to distribute his software quickly and painlessly, Michael invented `uucp`, thereby begetting a whole global network. With Brian Kernighan, he was responsible for the UNIX computer-assisted instruction software, `learn` (V7). Without the insight of Michael Lesk and the popularising touch of Brian Kernighan, the UNIX system would not have acquired the extroverted personality that commands such widespread loyalty.

* John Lions (Australia), University of New South Wales.

When John received 5th Edition UNIX it was *not* a trademark! The original license was dated 15th December, 1974 and the arrival of the tape/manuals proved to be a timely Christmas present. In 1976, John wrote "A Commentary on the UNIX Operating System" and formatted the accompanying V6 11/40 source code book. He said that `nroff` yielded some of its more enigmatic secrets so reluctantly, his gratitude was indeed mixed. However, without John's book, many of us would not remember that UNIX was once less than 10,000 lines of code and transportable in a student's brief case. And perhaps that immortal comment on line 2238 might have escaped us:

```
/* You are not expected to understand this. */
```

* John R Mashey (USA), Mips Corporation.

Well known for his hardware, software and CPU design skills, John was a co-author of the BSTJ paper "The Programmers Workbench". During the near 10 years spent at Bell Research Laboratories, he also worked on command languages, text processing, computer-centre UNIX issues and various features of V7, followed by management of applications projects and exploratory projects with bitmapped displays and programming environments.

* Jun Murai (Japan), University of Tokyo.

Architect and designer of the Japanese UNIX Network (JUNET) which uses TCP/IP protocols over dial-up telephone lines, Jun has a hard time explaining that his name really is Jun and JUNET is not named after him!

* Sam Leffler (USA), PIXAR.

Cutting his teeth in Lucasfilm, Sam has recently been grappling to produce a public domain version of Sun's NeWs system which means he's one of those rare breeds who has a fair amount of experience with PostScript. Of course, Sam has continued his interest in user interfaces and graphics systems by working on a 3-D modelling and animation.

* David L Presotto (USA), Bell Research Laboratories.

Dave tamed networks! His upas brought some order to a Babel of mail addresses and his IPC primitives provided a common basis for communication and remote file access via Internet, Ethernet, and Datakit.

* David Turner (Great Britain), University of Kent

Designer & architect of the functional programming language Miranda. David's research has primarily centered around functional programming languages. Currently, David is a professor in computer science at the University of Kent at Caterbury.

* Sunil K Das, the Conference Programme Chair and Chairman of The UK UNIX systems User Group (UKUUG), has borrowed freely in the above descriptions (thanks in particular go to Doug McIlroy). Any factual inaccuracies or misleading statements are unintentional but he acknowledges complete responsibility.

On Monday 11th and Tuesday 12th April, some outstanding one day tutorials will be taught by leading technical experts. From AT&T, Steve Buroff and Curt Schimel will present "Introductory" and "Advanced System V Internals" on successive days. In the final stages of planning are tutorials on "BSD Device Drivers", "C++", "Advanced System V Administration", and "Intermediate C".

A number of other tutorials are being prepared and further announcements will be made in the booking leaflet due to be distributed in December, 1987. Subjects under consideration include "Sendmail", "News", "Postscript", "X Windows", and "Software tools with Yacc and Lex".

It is intended to offer excellent discounts to those who book more than one tutorial and/or to those who book early.

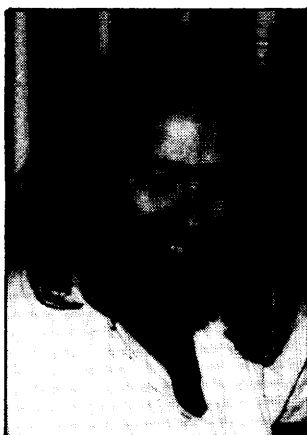


An anonymous well-wisher sent a 10th anniversary cake
to Owles Hall.

The EUUG secretariat appreciated it greatly,
thank you.

What's up with EUUG

Keld Simonsen
keld@diku.dk



EUUG Executive Committee

What is happening to the EUUG, and what is going to happen in the near future? Quite a few things, as you will see...

1. Financial History

In 1985, the EUUG suffered its first real loss. This caused us to focus on the problem of proper financial control and planning. It became clear that the group could not rely on conference income for survival.

Due to completely new budgeting practices in both normal running and conferences, the EUUG now has a prudent reserve. The finances of the EUUG are planned to break even. Due to the policy that a conference should not make a loss, a conference normally results in a small surplus. With the level of turnover for each conference, it is not realistic to plan for break even; we will either see a profit or a loss. Naturally, the former is preferred, but every care is taken to ensure that this surplus is not excessive.

Most of the excess from conferences remains with the group as a prudent reserve. This could easily be needed in the event that a conference or other activity is not as successful as planned. Suppliers will not accept the EUUG as a credit worthy body, and hence it is essential that sufficient cash exists at all times to fund the activities.

The group plans to retain between six months and one years operating expenditure in reserves.

This year, the Governing Board suggested that some of the reserves should be used to fund special activities:

- | | |
|---|---------|
| 1. A strategy workshop with all national groups | £10,000 |
| 2. A study on the EUnet network | £20,000 |
| 3. Promotion of EUUG and its 10th anniversary | £16,000 |

In the following paragraphs we discuss these activities further.

2. *The Paris Workshop*

The EUUG Paris workshop was held in order to improve the relations between the national groups and the EUUG executive committee, and also between the national groups themselves. Now that most national groups have grown really big, there are many of activities going on on the national level, and the groups could surely get inspiration from other groups, in many cases they share the same problems.

About 50 people from all the 14 national groups and the EUUG executive committee met in the beginning of September at a small Chateau outside Paris. We spent a weekend discussing topics such as strategy, conferences, networking, administration, newsletters and external relations.

The workshop was a forum to generate new ideas for the EUUG and the outcome is to be formalised in voting points to be put to the EUUG governing board. Already two weeks after it we had such a governing board meeting at the Dublin conference, and some important proposals from the workshop were approved there.

The governing board decided that extra copies the newsletter should now be available to institutional members at cost price plus 10%. A side remark was that the newsletter seems to have really improved under the hands of the new editor; the contents are interesting and the issues appear on time. So there is now value for the money. It is likely that this price will be of the order of £25, but the committee feel it is appropriate to collect the data from setting and mailing one more edition of the new newsletter before proposing the price.

This new price will make it possible for individuals who are employed by an organisation which is also a member of EUUG to have their own copy of the newsletter at an economic rate. As employees of an organisational member are allowed entrance to EUUG conferences at membership costs, the services obtained in this arrangement would almost be the same as being a true individual member.

This decision was initiated by a heated discussion on the student membership fee, which concluded in the above proposal, with students just fitting in like employees of an organisation (in this case their university). The many good reasons to subsidise students were mentioned, and the EUUG will also in the future have specially low conference fees for students, as the student conference grant scheme open to all students will proceed. At our last conference in Dublin about 10 students attended on grants, in contrast to almost none in the previous years.

Another decision of the governing board was that every national group should allow a member of another national group admittance to their national events on the same terms as their own members. The national events should also be better advertised to the members of the other national groups.

A lot of other good proposals were discussed in the workshop, some have already been decided on, and the EUUG executive is working on shaping the rest to make them workable proposals.

There is a fuller report on the Paris meeting elsewhere in this newsletter.

3. *EUnet and EEC.*

The EUnet study is to be done together with the EEC commission to investigate how EUnet could use standard ISO OSI protocols (including X.400 based mail) in all or parts of the network.

There is a possibility that the EEC commission will provide a big sum of money for computer equipment, communication lines etc. to implement this change. This should also apply to non-EEC countries, and the plan is to start in 1988.

The EUnet attitude to run OSI protocols is very positive when looking at the communications between the backbones, but the current cheap connections to the end machines have to stay as they are, and the operation of the network is not going to be disturbed during these changes.

It was felt by the EUUG executive committee that it was necessary for the EUUG to embark into this project in its role as a coordinator of EUnet. This work with its financial commitments would probably not have been undertaken otherwise, and a great opportunity to get proper funds for the network would have been lost.

4. EUUG 10th anniversary.

The EUUG 10th anniversary has been promoted in computer periodicals in most of the member countries. There was a splendid 10th anniversary reception and folkloristic dinner in Dublin (sponsored locally and paid by the conference, respectively), and we will go on celebrating the anniversary at the London conference in April 1988.

The London conference will also boast such interesting speakers as present or former Bell Labs people: Peter Weinberger, Mike Lesk, Dave Presotto, Doug McIlroy, Maurice Bach, and John Mashey, each of whom has made a significant contribution to the evolution of UNIX.

There will also be a tutorial programme on a high technical level held by world leading experts. The tutorial titles planned include: Advanced System V internals, BSD Device Drivers, Advanced System V Administration, C++, Intermediate C, Sendmail, News, X-Windows and Postscript. It is surely going to be an event of high quality, which you should not miss.

Future conferences will take place in

- Hanover, Germany in September 1988;
- Brussels, Belgium in April 1989; and
- Vienna, Austria in September 1989.

5. We need the EUUG

One general feeling on the Paris workshop was that we need the EUUG. We need the coordination role of the EUUG for the national groups. We all belong to an European UNIX community and have a lot of the same problems and attitudes.

The EUUG provides a forum where we can exchange these views, for example in the newsletter or at conferences. The EUUG can coordinate important activities like EUnet. The EUUG can provide excellent conferences and tutorials on an international level. This might be hard to do on your own as a national group, but together we can achieve it.

Fun With Spaces in TROFF



Jaap Akkerhuis
jaap@cwi.nl

Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands



1. *The Set Space*

Probably one of the most neglected troff requests is the `.ss`-request. To revive your memory, the form of this request is `.ss N`, which will set the space-character size to $N/36$ ems. The default value is (nearly always) $12/36$ th of an em. This request works in the current environment, and when there is no argument, the request is ignored. It is always ignored in `nroff`. The space-character size is the minimum word spacing in adjusted text. It is also the space you will always get when the text isn't adjusted. This is known in the typographical world as the *set space*. It also sets the width of the unpadding space character `"\ "`. The widths of the $1/6$ em narrow `"\|"`, the $1/12$ em half-narrow `"\^"` and, of course, the digit width space character `"\0"`, are not affected.

The set space is dependent on the font size, so that's why it is dependent on the "em". The size of the set space might vary between the different typefaces; some typefaces require a different value, depending on the design. Obviously, a constant width font wants to have a set space the width of normal characters, so that when the text isn't justified — adjusted in troff terminology — it would appear to have been printed on a typewriter. Also, you may want to change it depending on the style of the lay-out. Apart from this use there are more reasons why you might want to play with the `.ss`-request.

1.1 *Widows and Orphans*

It happens all the time: you have written your article or book, keyed it into your favourite machine, formatted it with `troff`, and are not really happy with the output. After some advice from a neighbour or reading a book about typography, the mess looks somewhat more consistent, but there is still the problem with the orphans etc. The word at the top of page 24, just before the opening of section 3, should actually belong on the bottom of page 23. Of course, one way to deal with this is to rewrite page 20 to 23, so it will fit, but then the opening of section 3 suddenly appears on the bottom of page 23, yuck! You can of course force this to page 24 by applying a `.bp-`, or better a `.ne`-request, but then you have this enormous white space left on page 23. Another way to fix this problem is to use the `.ss`-request. Around half way through the text that is going to be page 23, you make the set space somewhat smaller than the default value, and this will be just enough to pull the offending text from page 24 onto page 23, and the difference in spacing is hardly noticeable.

Of course, don't forget to reset it to its previous value so you can play the same tricks in other places.

1.2 *The Late Paper Problem*

A general rule is that papers will always be finished too late. But after it's finally finished, there is always another mistake to be found the moment the errand boy is around, waiting to bring it to the printer. A double word is found on the last page. Pushing the complete 200 pages again through troff on an overloaded machine will be too time consuming. An easy fix — when the offending word doesn't take a lot of space — is to typeset just the offending line and force it to be spread over the used line length using the `\p` escape. If there is a word missing, the set space is first reduced, before forcing the justification. Then the output is glued on top of the offending line. If the errors are in rather big words, you probably need to typeset a couple of lines with changed set space.

Of course, be prepared to do everything again, with your luck you will find that some coffee will accidentally drop over the galleys.

1.3 *A Large Number*

Your nephew pops up and wants to print the largest known Mersenne prime number in his school paper. Since this number is $2^{216091}-1$, he noticed that his calculator didn't grok it. Also, he already assumed that typing in 65050 numerals on his IBM-composer would probably not go without any error, so could you please help him.

Calculating the number is not a big deal and of course you keep the answer somewhere in a file. Feeding this to troff gives a problem, "Word overflow". After grumbling "why there are always these weird arbitrary limits in UNIX programs", you will realise that troff doesn't hyphenate the number and that you don't want it to be hyphenated anyway; it wouldn't look right.

Apparently you have to break the number in smaller units and glue them together. Breaking the number can be done with your favorite editor which can handle arbitrary large files or just a little C-program, replacing each numeral for itself and a space, and inserting some newlines as well. Then you set the setspace to the smallest possible value, 1/36th of an em.

Now you notice that the last line of the block of numerals has less space than all the lines before. This is fixed by making the line length exactly to the length of the numerals and spaces you want it to have, so the spaces are not stretched. As an alternative, you can of course set the lot not left justified. So this will get you something like:

```
7460 931030646613436 8733957940051148954022875408497732880511330497779366272527
87806643956351409557300083644941548827574272300629992209408195687757874506481
15775343181409425298589715882838808681344634512372399168539091687963336
```

... lots of numerals omitted

```
44030338221036632627130427722612194544556452516799417964416162136915976643526
40545872469131519545069120183118538411805217750684693278676451411187769133620
3815528447
```

If you really get carried away by this problem, you can edit the ditroff output as well to get rid of that little space. Of course, that will make your line length somewhat shorter, but who cares, it's for your nephew after all.

2. *Constant spacing*

The constant spacing request, `.cs F N M`, will set every character of the font *F* with the width of $N/36$ th em. If the optional *M* is given, the em is *M*-points. The characters are centered in this space, even if this character is wider than the

available space.

This request suggests that you can use this to generate a constant width font.

These lines are typeset with
.cs R 16.

There is even spacing for the characters, but as you can see,
the result is an insult to the eye.

The result depends a bit, of course, on the variety of widths in the font. But it is much better to use a typewriter style font if you want to simulate a typewriter. Even then you might want to use the .cs-request, with the constant width set to the width of the typewriter font, in case you use a character which isn't in that font. It will then get it from the special font (S) when mounted. With some luck the result might not be too bad.

2.1 Watch that man

Although it appears that the constant width request is quite useless, it can come handy in some cases. It might happen that after an EUUG conference somebody sends you data like this.

```
0x0000,0x1FA0,0x0000,
0x0000,0x7FFC,0x0000,
0x0000,0xFEFF,0x0000,
... etc
```

The data is actually describing a 48 by 48 pixel bitmap. Apparently a picture has been taken of you, but showing this hexadecimal stuff won't please the rest of the family. So how do you turn these numbers back into your usual self?

To put it on a line printer, you can either write a C-program or use the next sed-script.

```
# icontox

sed '
s/0x//g
s/, *//g
s/0/ /g
s/1/ #/g
s/2/ # /g
s/3/ ##/g
s/4/ # /g
s/5/ # #/g
s/6/ ## /g
s/7/ ###/g
s/8/# /g
s/9/# #/g
s/[Aa]/# # /g
s/[Bb]/# ##/g
s/[Cc]/## /g
s/[Dd]/## #/g
s/[Ee]/### /g
s/[Ff]/####/g
' $* | sed 's/ *$//'
```

Applying this script you can get lines which look more printable.


```
##### #
#####
#####
#####
#####
#####
#####
```

To turn this into something nice, you need to work somewhat harder. This is what I used for this paper.

```
.nf      \" nofill mode wanted
.nr y \n(.lu/2 \" get half length of line
.ps 9    \" pointsize 9
.tr #.   \" translate # to . on output
.cs R 3  \" constant spacing
.vs \w' 'u      \" make the vertical spacing equal to horizontal
.in \nyu-\w'#####'u \" center the face
```

Output of `icontox` comes here

```
\" And now restore everything to normal
.cs R
.tr ##
.ps
.vs
.in
.fi
```

After doing all this you will get something like the next picture.



As you can see, the result can be shown to the rest of your family. You might want to play with the constant spacing factor or pointsize a bit, to get the best result, since this depends on the size of the full stop for your output device. Also, if you want to enlarge the picture, you can just bump up the pointsize and the constant spacing factor.

2.2 Bitmap fonts

This technique suggests that it is easy to include screen dumps from your favourite bitmap screen in your paper. It turns out not to be that easy. A locally written `screendump` program family for "blit terminals" (or DMD5620) has been converted to give its output in a format to be processed by `troff`. To prevent the "Line overflow" message, `troff` had to be adapted. Apart from that, it takes considerable time for `troff` to process a bitmap created like this: about 820 CPU seconds for a blit screen.

The usual way to include screen dumps is to pass them through `troff` using the `\l` escape or the `.cf-request` (for modern versions of `troff`) and to do major surgery on the `troff` output. The problem with that is that `troff` will lose control of its output, since it doesn't know what magic is going afterwards. A more elegant way of handling this would be to have a font available on the output device which describes pixels.

Consider a character set, where each character describes 8 pixels. So the font will contain 255 different characters of the same width. For displaying the bitmap off a

1K*1K screen you only need 1024 lines of 256 characters. The input to troff would be something like:

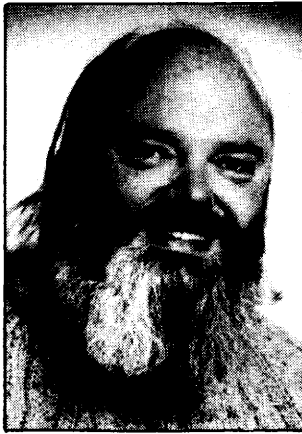
```
.nf      \" nofill mode
.ft Bm   \" select bitmap font
Ab(es\(*A\ - - \" weird bitmap font character mapping
.fi      \" restore fill mode
.ft      \" and font
```

The moment that you want to have a dump of a colour screen, you are of course lost, but as long as there is no way to directly print colours without colour separating, it is not a real problem. Get the screen dump in black and white, get your camera out and take that picture. The print shop will do the colour separation for you as usual. If you really want to be fancy, you can have the screendump program do the colour separation.

2.3 Acknowledgements

The idea of abusing the set space request for patching up papers on the last moment comes from Bob Garufy. Rob Pike delivered the bitmaps and the icontox program. Paul Vitányi inspired the "large number problem". Jim MCKie made the local screendump program family. Paul Klint suggested the idea for the need of a bitmap character font. Peter J. Weinberger permitted the use of his picture. Carol Orange assisted in turning this article into (American) English.

C++ Gossip



*John Carolan
john@puschi*

*Glockenspiel Ltd.
Dublin, Eire*

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

By the time you read this, the C++ workshop in Santa Fe will be history. Anyway, I think the list of papers is interesting, just to show what various people are doing with C++:-

Steve Dewhurst, AT&T	The Architecture of a C++ Compiler
Michael Ball, TauMetric Corporation	The Oregon Software C++ Compiler
John Carolan, Glockenspiel	C++ for OS/2
Ken Friedenbach, Apple Computer	Porting C++ to the Macintosh OS
Philippe Gautron and Marc Shapiro, INRIA	Two Extensions to C++: a Dynamic Link Editor and Inner Data
Jonathan Shopiro, AT&T Bell Labs	Extending the C++ Task System for Real-Time Control
Tom Doepfner and Alan Gebele, Brown University	C++ on a Parallel Machine
Roy Campbell, University of Illinois at Urbana-Champaign	CHOICES — A Multiprocessor Object-Oriented Operating System

Oliver McBryan,
New York University

Dave Detlefs,
Carnegie-Mellon University

Steve Mahaney and Ravi Sethi,
AT&T Bell Labs

John Rose,
Thinking Machines, Inc.

Kevin Kenny,
University of Illinois at Urbana-
Champaign

Peter Kirslis,
AT&T

Kenneth Brown,
Cardinal Information Systems

Judith Grass,
AT&T

James Coggins,
University of North Carolina

Michael Tiemann,
MCC

Dan Schuh,
University of Wisconsin

Mark Rafter,
Warwick University

Bjarne Stroustrup,
AT&T Bell Labs

Keith Gorlen,
National Institutes of Health

Ken Fuhrman,
Ampex Corporation

Tsvi Bar-David,
AT&T

Al Conrad,
University of California at Santa
Cruz

James Coggins,
University of North Carolina

Jim Waldo,
Apollo Computer Inc.

C++ Environments for Parallel Computing

Avalon: C++ Extensions for Transaction-
Based Programming

Concurrency and C++

C*: a C++ -like language for data-parallel
computation

Encapsulators -- A Metaphor for Software
Organization in C++

A Style for Writing C++ Classes

Inc., Object-Oriented Databases

Using C++ in Language Processors

A Summary of C++ Applications under
Development at the University of North
Carolina

G++: A Free C++

Parameterized Types for C++

Extending Stream I/O to include Formats

What is 'Object-Oriented Programming'

OOPS: A C++ Object-Oriented Program
Support Class Library

An Object-Oriented Class Library for C++

Teaching C++

Modelling Graphical Data with C++

Integrated Class Structures for Image
Pattern Recognition and Computer Graphics

Using C++ to Develop a WYSIWYG
Hypertext Toolkit

Mark Linton and Paul Calder,
Stanford University

The Implementation of an Object-Oriented
User Interface Package

Mark Linton,
Stanford University

The Design of the Allegro Programming
Environment

Ragu Raghavan,
Mentor Graphics

A C++ Class Browser

Tom Cargill,
AT&T Bell Labs

Pi: A Case Study in Object-Oriented
Programming

Only four of the speakers are actually travelling from Europe, although Olie McBryan is a native of Dublin and Bjarne Stroustrup is from Aarhus. I will include a report on the workshop in the next newsletter.

I have not had many replies to my question in the last newsletter about which colleges are teaching C++. If you are teaching it, please mail me at

mcvax!clitelcpuschi!john

and I will print a list in a future newsletter.

I heard there was a lot of interest in C++ at the recent OOPSLA conference in Florida. If anyone in EUUG was there, please can they write something for this column!

Technical tip

If you would like to make a really abstract class, i.e. one where there is no hint at all about what kind of data is being represented, you can do it like this.

In `foo.hxx`, the file which declares `Foo`:-

```
#ifndef IMP
typedef void* IPTR;
#else
typedef FooImp* IPTR;
#endif

class Foo
{
    IPTR p;
public:
    Foo();
    ...
};
```

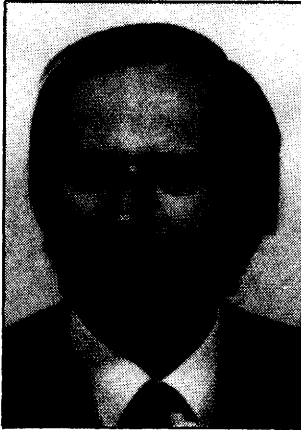
Then, in `foo.cxx`, the file which implements `Foo`:-

```
struct FooImp
{
    // anything you like
};

#define IMP
#include "foo.hxx"
```

As far as the user of `foo.hxx` is concerned, `FooImp` is invisible.

The X/OPEN Mid-Term Report



John Totman

*ICL
Bracknell, England*

John Totman is an electronics development engineer who has been involved in the engineering support, development and marketing of operating systems since the early 1970's.

He more recently strayed into UNIX when managing the development of commercial applications for departmental users.

A convert to the cause of standards and applications portability, John is currently a Marketing Manager for X/OPEN.

X/OPEN COMPANY LIMITED

After an engagement of nearly three years, the members of the X/OPEN Group finally "tied the knot" on the 10th of September this year. In short, X/OPEN has been incorporated as a limited company, with existing members as equal shareholders.

The Benefits of the Incorporation

This now means that X/OPEN will be managed as a truly independent company, with its own dedicated staff and offices in both the UK and the USA. The President and Chief Executive Officer for the newly formed X/OPEN Company Limited (the company has been created under UK law) is Geoff Morris who was previously the ICL X/OPEN Strategy Manager and Chairman of the X/OPEN Strategy Committee.

Other formal appointments include James R. Bell of Hewlett-Packard, as Chairman of the X/OPEN Board of Directors, with fellow strategy managers from AT&T, Bull, DEC, Ericsson, ICL, Nixdorf, Olivetti, Philips, Siemens and Unisys all becoming members of the X/OPEN Board.

Although X/OPEN has always been independent of any single vendor, incorporation has made independence much more tangible. Becoming a legal entity will enable X/OPEN to consolidate its corporate role as a focal point in drawing together the industry on the standards issue. In addition, incorporation will make it easier for non-members to participate in X/OPEN: already plans are well advanced to establish formal channels direct to users and ISVs in order to gain their contribution to the development of the X/OPEN Common Applications Environment (CAE).

X/OPEN Technical Reports

The subject of contributing to the CAE leads me to two topic areas where X/OPEN has completed major technical studies this year: On-Line Transaction Processing and a

Transport Interface Definition (XTI).

Formal reports for both these topics have been published as "White Papers" for review and comment by other standards groups and major industry organisations. In summary, these reports cover the following subject matter:

On-Line Transaction Processing — A Reference Model

This report assesses the applicability of the UNIX operating system as a vehicle for the support of a distributed Transaction Processing environment. In particular the report explores the requirements of data integrity, scheduling to ensure predictable response times, and flexible terminal support.

The concepts and terminology applicable to Transaction Processing systems are clarified as working definitions, and set out as a proposed layered functional model for a distributed Transaction Processing environment.

The good news is that the detailed development of the reference model has led to a number of well argued conclusions which support the use of UNIX as a vehicle for a Transaction Processing environment. However, there is a need for some basic functionality to be added to the UNIX kernel: guaranteed output to files and concurrent input from peripherals in particular.

The next phase for X/OPEN, following discussions with vendors, software houses and other interested parties, is to develop a complete X/OPEN TP Interface Definition for incorporation within the X/OPEN Common Applications Environment.

X/OPEN Transport Interface (XTI)

The XTI report defines a transport service interface which is independent of any specific transport provider. The interface is provided by way of a set of library functions for the C programming language. XTI serves three main purposes:

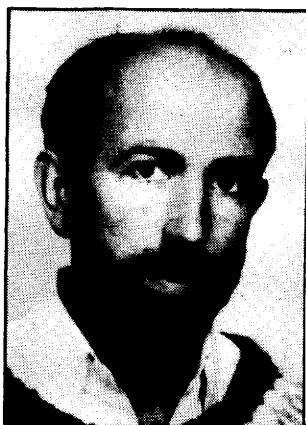
- XTI describes a wide set of functions and facilities together with a minimum workable subset that enables applications portability across systems incorporating XTI.
- XTI is concerned primarily with the ISO OSI Transport Service Definition (Connection-Oriented or Connectionless). However, it is adaptable to other types of provider. In particular XTI is extended to include TCP and UDP.
- XTI is UNIX-version-independent.
- Once again, X/OPEN is seeking to elicit comments on this draft definition from third parties for inclusion in the final definition.

A White Paper Christmas?

Future topics to be covered by X/OPEN white papers include Security and XPPC, with other subjects coming down the line as well.

A Merry Christmas from the X/OPEN Company.

EUnet



Peter Houlder
uknet@ukc.ac.uk

*Computing Laboratory,
 University of Kent*

1. Introduction

This is the third of a series of articles giving information about the European UNIX network, EUnet. The credit for the main part of this article goes entirely to Yves Devillers of INRIA, the French backbone. The end bit contains some notes on mail addressing written by yours truly. The articles supplied so far by the backbone sites are keeping this column going, so THANK YOU and please keep up the good work. However, the continuation of this column depends on articles, paragraphs, notes and ideas from anyone, not just backbone people, concerning aspects of European networking. They will all be gratefully received (please mail the above).

2. FNET — Yves Devillers (*inrialdevill*)

Some 90 sites (mostly educational and public research) now constitute FNET, the French EUnet sub-tree, around its backbone at INRIA (National Institute for Research in Computing and Automatics, a public research institute funded by the French Ministry for Industry). Until very recently sites were connected to the backbone (or between them, but this is out of the scope of FNET) using PAD to PAD over the public X25 network (sometimes called reverse PAD), thereby limiting access to medium and large sites (X25 fixed monthly rental is about 290 to 430 UK £. We also had a fully homologous V25 ACU (with battery saved memory and a real K7 player for outgoing excuse message!) for few dialled passive sites. Very recently low cost ACU began to be homologous, allowing access to smaller active sites.

INRIA, as a research institute, is involved in European research networks and is participating actively in "Aristote", the French X400 research network.

We have two research networks in France (three if we include FNET), the second one, "Reunir", being promoted by Ministry for National Education and by Centre National de la Recherche Scientifique, and presently based upon EARN. As well as INRIA, CEA (atomic energy research), CNES (space agency), EDF (electricity company) and CNET (PTT's research centre) are also members of Aristote. Members of Reunir are usually big computing centres (IBM 3090, Cray, Multics...) with lot of physicists and numerical modelling, whereas FNET are the classical UNIX people running on medium (VAX 780) to low size (Sun 3, SPS7) computers.

Equipment has been allocated by INRIA to run gateways between Aristote and other networks (CSNET and FNET, EARN being at early stage of negotiation). That

equipment consists of two BULL SPS7/70 (68020, 8Mb, 800Mb, ethernet and X25, System V plus Berkeley IPC, DoD suite and ISO session interface, X400 as native mailer). One computer is fully dedicated to the FNET backbone and should be in order at the end of this year (freeing the VAX 785 of all the incurred overload). Two INRIA employees are spending part of their time on FNET administration and relations with other networks and two students are working half time on network maintenance and billing. Starting next year FNET members will be asked for a yearly fixed contribution (290 UK £ for educational, 870 UK £ for commercial) to pay part of these student costs.

Last May we contributed a lot to a special issue on communication in a UNIX newspaper dedicated to SPS7. The immediate result was a good number of new small sites joining the net, with too many of them asking for news! This is our real problem in FNET with only 16 sites receiving news, and 30 asking for it. INRIA is retransmitting news to 12 sites, one of them (crcgel) retransmitting it to 3 other sites, but it seems very difficult to have a leaf node act as a redistributing site (we call them T-bone) for news, and the present 30Mb monthly volume (steadily doubling every 10 or 12 months) does not help to encourage volunteers.

AFUU, our local UUG, is pushing us to have more sites and more users on the net (and thus more AFUU members). We are convinced we cannot provide any kind of service (setting up node, debugging communication problems...) to end users as all our time is already spent running the backbone. A few users, exclusively from public research institutions, get access through minitel (French videotext) to their mailbox on a dedicated computer. This service is not yet very developed; it could be a good solution to AFUU members wishing to access the net, even though they do not have sufficient manpower to maintain a full mail node. AFUU will eventually be running such a node.

FNET people meet together very informally 4 to 6 times a year in the "AFUU-reseau" (reseau is the French word for network) working group, J.C. Petithory being the animator of it. There we enjoy talks on networking and related subjects. I should say "enjoyed" as most (all?) the meetings in the last 18 months have been dedicated to evolution within the net, switching to domain addresses, tariffs, migrations, problems at the backbone and so on. We also intend to make a tape available for FNET members, including communication stuff, configuration files and all the associated goodies. That tape should be ready for the annual AFUU meeting.

Activities in the near future, at the backbone, will be devoted to new links testing (switched numerical telephone at 64kb/s) or setting up (leased line toward Holland, IP on top of X25 through Sunlink or SPS7) and experimenting with X400 mail using mailway, a mailer channel for sendmail. Activities within FNET will include the final phase of migration toward domain based addressing, solving news problem, thinking about a possible status for the future FNET and having good talks at AFUU-reseau meetings.

3. Addressing

Forming addresses is always a lively problem, so I've divided this up into 3 sections: addressing where all sites are domain based; addressing where all sites are uucp based, and mixed addressing.

3.1 Domain based addressing (733)

All these examples assume an international standard format, the "Little Englander" mentality, that created UK-style addresses is handled later.

Mailing a site to which you are directly connected is simply:

user@site.sub-domain.domain

where sub-domain may be repeated zero or more times. So we at ukc might mail *user@alvey.uk*, *user@ed.ac.uk* or *user@stl.stc.co.uk* (where "alvey" shows no sub-domains, "ed" is in the "ac" sub-domain and "stl" is in the sub-domain of "stc", which is in turn in the "co" sub-domain of the "uk" domain). When explicit routing is required, then the following form of mailing can be used:

user%end-site.sub-dom.dom@rout-site.sub-dom.dom

where *end-site* is the site you want the mail to end up at, and *rout-site* is the site through which you are routing the mail. All the routing site then does is strip off the bit after the @, finds the last % in the reduced string, changes it to @ and forwards the mail on to the site after the new @. By this method multiple explicit routing is supported, hence mail to *user%d%c%b@a* would arrive at "a", who would then forward the mail to *user%d%c@b*, who in turn would forward it to *user%d@c*, who would finally send it to its end destination, that is *user@d*.

In the UK (and apparently in New Zealand), things are done differently. Hence on our academic network, JANET, site *uk.ac.foo* would mail site *uk.ac.bar* as *user@uk.ac.bar*, i.e. the domains and sub-domains are reversed. We at ukc, being an international gateway, do things properly, so we first try to make sense of things assuming a mailer has followed international standards, then we domain flip everything to see if a reversed address makes sense. This does, however, cause some amusing problems. If we mail *user@uk.ac.rl.de*, then the mail does not go to machine "de" at the Rutherford Laboratories in the "uk", but instead it goes to Germany, because we check every address before reversing it, and "de" is the registered West German domain. Basically, UK users mailing abroad through us should think internationally.

3.2 UUCP addressing

Uucp sites address each other with "bang" style addresses, (that is addresses separated by exclamation marks "!"). Hence, site "a" mails a user at site "b":

bluser

Explicit routing is easy, as routing gets added to the front, so "a" mailing "d" via "b" and "c" would mail:

blcldtuser

Site "b" then strips off his site name and the next "!", and forwards it to site "c", and so on. In this case addressing is in the reverse order to the explicit routing for domain style addressing. It is this difference in approach that causes the "fun" when domain base sites connect to uucp sites.

3.3 Mixed Domain and UUCP style addressing

Mixed communication between domain based and uucp based sites, where explicit routing is involved is something of a minefield. There are enough difficulties in routing the mail in the first place, but the really exciting problems occur with the formation of a meaningful reply address for the recipient, who might understandably want to answer his or her mail.

Any articles detailing these problems, what caused them and ways of getting round them would be welcome, but a few examples that we at ukc have heard about, or encountered, might start the ball rolling. Examples with explanations, as with the five below (my thanks to Sean Levisaur), can be both illuminating and (hopefully) amusing for everyone.

1. Domain based mailer at "c" gets address a%b@c, removes the "@c" and tries to mail "a%b". That is an easy one, but it has happened.
2. A cc: field address of "a!" can occur, when a mailer decides to remove both the user and the site from an "a!user@ba" type address.
3. UUCP sites can often have fun with "a@b@c"
I've also heard of this: "a@b!c@d", although I haven't actually seen that one.
4. Finally a specific ukc one to show how we got things wrong, before we recognised the csnet domain. We once received an address:

user@csnet.tektronix

so we tried to decide where to send this. Our first try is to look for a "tektronix" domain. We didn't find that, so at ukc we domain flip and see the address:

user@tektronix.csnet

At that time we didn't recognise csnet, so we make our next try to see if we have a site of matching name and sure enough we find tektronix.uucp, so we send the mail to:

user@csnet.tektronix.uucp

thereby treating csnet as part of the uucp site called tektronix. The answer in this case was to recognise the csnet domain, and everything worked.

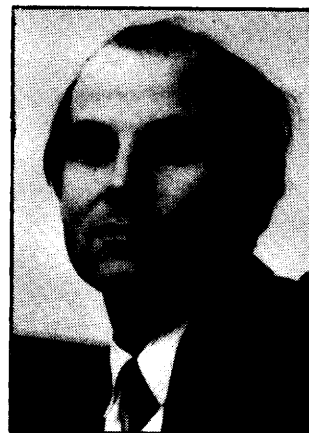
UNIX in the Marketplace

Dominic Dunlop
domo@sphinx.co.uk

Sphinx Ltd.
Maidenhead, England

Dominic Dunlop is the Research and Development director of Sphinx Ltd, a UK software distribution and services company he co-founded in 1983, after experience in supporting Zilog's range of super-micro computers. Sphinx centers its operations around non-proprietary operating environments, selling in a variety of third-party and self-written software products across hardware from name different vendors.

Dominic's current rôle is that of bringing complex new products into Sphinx' offering by first understanding the technical and marketing issues involved, then working to address them in the context of the company's current capabilities and activities.



I'm Dominic. Buy me!

Hello. As Research and Development director of Sphinx, a company which sells and supports software packages for UNIX* and several other operating systems, I get to research new developments. To put it another way, I'm the person who monitors all that neat stuff that all those clever people have dreamed up, just so that I can say why it won't work. A typical killjoy critic, in other words. Consequently, your gentle editor has asked me to write a column for your newsletter, commenting on what's hot and what's not in the application software world, and whether it's all just marketing hype anyway. Provided that I don't get sued, and that I do get prodded for delivery, this should turn into a regular feature. If there's any product area that you'd like to see discussed, let me know.

And this issue's topic is...

UNIX on the 80386 — Bloodshed or Watershed?

The issue of UNIX on the Intel 80386 has been the source of a lot of smoke, punctuated by occasional flashes of light, for much of 1987. It all started back in January at the Uniforum show in Washington, when Microsoft announced that Xenix† and UNIX System V, release 3 were to merge by mid-1988, resulting in a single operating system which would be able to run both *coff* (Common Object File Format) binaries generated for System V by AT&T's compilers and linkers, and *x.out* binaries produced for Xenix by Microsoft's C-merge compiler and tools. The trouble

* Yes, UNIX is a registered trademark of AT&T in the USA and (some) other countries

† Xenix, MS-DOS are registered trademarks of Microsoft Corporation

with the initial announcement was that it was hazy about AT&T's part in the development. This turned out to be because, although AT&T had been involved all along, it were not quite ready to announce its side of the story in January. AT&T got around to putting out a press release in February. By then, the air was full of speculation and misinformation about the effects of the move on Microsoft, AT&T, Xenix outlets such as the Santa Cruz Operation and IBM, UNIX outlets such as Interactive Systems, Microport and IBM, and — oh, yes — the poor old user.

Whatever you might think of the architecture and heritage of Intel's 80386 processor, it has several important commercial and technical properties — properties which, correctly used, could get over one of the bugbears which has increased the difficulty of developing, selling and supporting software for the UNIX operating system over the past ten years. This problem is that, while you can quite easily write source code which is portable across UNIX implementations, *binary* code portability is all too rare. Consequently, software developers waste time moving their application from one implementation to another, when they could instead be improving its functionality or performance. Worse still, since porting (particularly to Intel's previous processors) is one of those jobs which is 95% crushing tedium, while still requiring 5% of inspiration, it is too often delegated to technicians who can't actually supply the necessary inspiration, and who consequently produce products which, while they work, don't work well.

How can the 80386 get software authors off this treadmill? Well, here's what it's got going for it:

MS-DOS heritage: Say what you like (or dislike) about it, MS-DOS has spawned more software more quickly (and more profitably) than any other environment. Systems which run UNIX can't afford to ignore this vast range of products, or the enormous base of users it has generated. The 80386 has a "virtual 8086" facility which can be used to run multiple MS-DOS tasks under the control of UNIX.

PC heritage: Because of the success of the IBM PC family, lots of systems designers know how to build Intel processor-based value-engineered systems which can be cranked out at the lowest possible cost. Consequently, PC-class computers give more "bangs for the buck" than almost any alternative. The 80386 at last allows an unconstrained implementation of UNIX on this class of hardware.

Xenix heritage: The 8086 and the 80286 were not pleasant targets for a port of UNIX. In fact, a conspiracy theorist once described IBM's PC-XT to me as "the most powerful system that won't run UNIX." Despite this, Microsoft and the Santa Cruz Operation made it their business to create a port of Xenix which squeezed worthwhile multi-user performance out of XT-class systems. The result was one of the most obvious success stories of the UNIX marketplace: Xenix, now running mostly on 80286-based hardware, has the largest installed system base of any single UNIX implementation, and supports a wide range of application software.

Big segments: One of the most unpleasant aspects of the 8086 and 80286 was the 64k segment limit, which meant that programmers had to play all sorts of tricks (and put up with all sorts of compiler, linker and system bugs) in order to manipulate more data than could be held in \$10-worth of chips. This problem made it difficult or impossible to port big engineering, graphics or modeling applications to Intel-based hardware, so confining its utility to the low end of the

market. The 80386 still has segments, but, since they're four gigabytes long, addressability problems are a thing of the past (except for poor old MS-DOS, tied by an obsolete hardware design to just 640k of directly addressable memory).

Built-in MMU: The processor architecture which has spawned the most hardware designs for UNIX systems is that of the Motorola 68000. The sad thing about all these designs is that, although they use processors which run the same instruction set, they do not in general allow binary application portability. The main reason for this is prosaic: Motorola took its time in shipping a companion memory management unit for the CPU — and, when they did, designers didn't like it. As a result, different Motorola-based hardware designs have different memory-management schemes, and require different application base addresses, page sizes and so on. Even if operating system implementors could agree on system call identifiers and the like†, the hardware would still prevent binary portability. Recognising this weakness in its competition, Intel incorporated memory management hardware of adequate (although not superlative) performance into the CPU chips of both the 80286 and 80386. Systems designers, presented with a *fait accompli*, have little excuse to supplant this circuitry with a plethora of incompatible alternatives.

Reasonable power: The 80386 clocks in at about three MIPS. That's a reasonable amount of power to throw at the medium-sized operating systems that both the AT&T and Berkeley variants of UNIX have become. In fact, it's precisely because Xenix stayed as a relatively small operating system, with version 7 features at its heart, that it was able to squeeze so much performance out of 8086- and 80286-based PC-class systems, and find such a ready market among those who wanted a multi-user system for the price of single-user hardware. The 80386 holds out for Intel-based systems the prospect of breaking out of this low-end ghetto.

So, in summary, the Ideal Implementation for the 80386 would have the following capabilities:

- Ability to run existing binary MS-DOS programs (including the "ill behaved" ones which operate on the (largely correct) principle that the way to get the best from MS-DOS is to pretend that it's not there)
- Support for low-cost hardware
- Ability to run existing 80286 and 80386 Xenix binaries (as well as 80286 and 80386 UNIX binaries)
- Full 32-bit addressability, allowing easy support for large programs and modern bit-mapped interfaces
- Standardised memory management scheme (including paging, as support for that's built into the 80386 CPU too)

† Unisoft ports to Motorola architectures have long allowed portability at the level of unlinked object modules through this type of standardisation.

- The "big machine" features of UNIX V.3

Guess what. That's just what the merged product is going to do. What's more, at the PC (and PS) end of the market, many vendors will be competing to sell UNIX implementations which, as far as their ability to run any existing application goes, will be identical. This should result in price competition, and in product improvements as competing vendors attempt to make their particular product stand out by, for example, supporting more devices or being easier to administer. And it'll be called UNIX wherever it comes from, as AT&T has licenced use of its trademark to Microsoft. (I suspect that Microsoft and its agents had become very tired of trying to convince potentially large customers wanting to purchase UNIX that, even though Xenix had a different name, it was SVID-compliant and had been passed by the SVVS.)

Sounds wonderful, doesn't it? The snag is that it doesn't happen until the middle of 1988. Until then, programs developed under Xenix — even Xenix/386, the native 80386 version — will run only under Xenix, while those developed under V.3 for the 80386 will run only under V.3. For developers this means that, for the next eight months or so, either two ports must be done (and subsequently supported), or part of the potential market must be ignored. For example, although a Xenix port of a software product will satisfy many small business users, it may not impress a large organisation which has standardised on System V across a wide range of system types. Conversely, a user already running several Xenix applications is not going to be interested in a product requiring the purchase of a different and currently incompatible operating system. At least a port to either environment now should work in both after the merged product hits the streets — despite the reasons some suppliers are giving for staying with their implementation even after the merge.*

Still, I suppose having to do even two ports is a considerable improvement on the twenty or so it currently takes to address a sizable slice of the Motorola-based marketplace. In fact, the 68000 family, which had appeared to be the flavour of the decade for microprocessors running the UNIX operating system, seems to be in trouble, with many formerly loyal hardware suppliers turning either to RISC architectures to get more power, or rushing out 80386-based boxes in the hope of building sales volume.

What's my conclusion? Well, MS-DOS has shown users that binary portability on the level of "if it fits in the diskette slot, it'll run on your hardware" is achievable. To date, UNIX has not been able to deliver this sort of portability, and this has been one of a number of factors which have limited its acceptance. The 80386 and the agreement between AT&T and Microsoft fixes this deficiency, *provided that the 80386 becomes the dominant processor in the UNIX marketplace*. And it looks to me as if it will.

Footnote

It's a surprising fact that there does not appear to be an implementation of BSD-style UNIX for the 80386, even though the chip's architecture is entirely up to the challenge. Is this another indication that BSD's star is on the wane?

* A quick plug: Sphinx has a foot in both camps, distributing both Xenix/386 and UNIX V.3 for 80386-based PC-type systems

EUUG Tape Distributions



Frank Kuiper
mcvax!euug-tapes
mcvax!frankk
euug-tapes@cwi.nl
euug-tapes%cwi.nl@uunet.uu.net

Centrum voor Wiskunde en Informatica
Amsterdam

_] [_ _ | |
 <-----| -1
 0-0-0

In this issue of the EUUG Newsletter I will try to explain the bits and peaces about the EUUG distributions on 1/4" cartridge tapes. Following that, there is the standard list of available tapes, including a new one: EUUGD12 "Dublin, autumn 1987 conference".

The last couple of years, more and more manufacturers of smaller mini computers have supplied 1/4" cartridge streamer tape drives with their machines. The well known 1/2", 9-track reel tapes are no longer supported or considered to be too expensive.

Naturally, because of this, there has been a growing demand for software distributions on cartridge tape. It would of course seem logical to start supplying the EUUG distributions on these cartridges, but there are, as I found out, some snags.

First of all, there are several formats of writing (and of course reading) the data. They are know under names like: QIC-2, QIC-11, QIC-24, etc. The difference between these formats is the number of passes on the tapes. QIC-11 uses four passes, QIC-24 uses nine.

Secondly, not all of the hardware and software on the various machines interprets the data on the tape the in same way. Obviously there is the difference between the "Little Endians" and the "Big Endians" (byte order differences). Another difference is that at least one machine inverts bit polarity!

So, in order to see clear through all of this, I asked around for information. After a request from me on eunet.general, I received several replies, which I received in gratitude. Most noticeably I have to thank Jamie Watson. Most of the descriptions of the different formats were supplied by him.

In order to make life a little bit easier, I will start with the description of the Sun format. This is just because I have a Sun drive available. So all the tapes will be made on this drive. What follows after are suggestions on how to get other machines to read this format.

As there are several Sun formats used, I had to choose one. Our machine can read/write QIC-11 and QIC-24 formats, so it all came down to choosing one of these two. A quick survey of the replies I received, reveiled that about half of them mentioned being able to read QIC-24. The other half mentioned: QIC-2, QIC-11, QIC-36, QIC-120, 1/2" reel tape, and some "don't know"s.

So QIC-24 it is then. As this is a first try in using this format for distributions, I would appreciate it if anyone who orders these tapes and is, or is not, able to read the data would send me a small note about it. Both e-mail and surface mail addresses are listed in this article.

To start with the "standard" QIC-24 format. This is a format that writes in 9 passes on the tape. The following machines can read this format without any necessary conversions, but they might have some details to be looked after:

Sun-2 systems

Not all though. Only those made after a certain date (sorry, don't know any details).

Sun-3 systems

Altos 2086 & 3068

Convergent Technologies MinFrame and MightyFrame

Motorola 8000 series

Arete

AT/Xenix

This one seems a bit sensitive to block sizes. Try to dd the data from tape first.

IBM RT/PC

Same as for the AT/Xenix

Apollo

Here I have a problem. As this machine seems to be very sensitive to block sizes, the suggestion was to write the tape using a blocking of 1! This means that for certain distributions numerous tapes are needed. It is of course silly to do this and let everyone else pay for it. If anyone with this kind of machine can't read the tapes that I will send (they will have a blocking of 10 or 20), then they will have to contact me again so that we can work something out.

Integrated Solutions

This one normally uses a 512 byte header for each 10k byte block. So, to read a Sun QIC-24 tape, you somehow have to include these...

Confused? You will be after reading the following list of machines that need magic tricks...

Sun-1 and some Sun-2 systems (before a certain date)

These only read QIC-11.

Heurikon

Uses QIC-11

Silicon Graphics

Uses QIC-24 à la Sun, but has it's bytes swapped.

NCR Tower

Byte swapped QIC-24.

Sperry 5000 series 20, 30, 40 and 50's

These are actually NCR Tower machines, so they use byte swapped QIC-24.

PCS

Uses QIC-11.

Plexus

All models read QIC-11, but...

P-25, P-35, P-40, P55, P-60, P-75 and P-90 types: these are almost the same as the QIC-11 Sun format, but they have the bit polarity inverted! This means that whenever the machine reads a 1 bit from the tape, it thinks it is a 0 bit! And vice versa. (How weird can you get?)

Plexus invented a new option for their dd command on their System V release: `conv=flip`. This corrects the polarity.

The other types (P-15 and P20) use the normal polarity. So, they are compatible with the Sun QIC-11 format.

HP 9000 series (200, 300, 500 and 800)

These use a format of their own, not compatible with any QIC format.

AT&T 3B2

As for the HP, they use already formatted tapes which are not compatible with the rest of the world.

MicroVax

Completely different cartridge.

I hope that all this does help you a bit. If you have any questions, or even better, suggestions on this, please let me know.

Two last remarks. As for the other tapes, the cartridges will be written with tar, using 20k byte blocks. If this fails too often, I will consider a different format.

As these cartridges are more expensive than 1/2 inch reel tapes, this format distribution will cost more. A cartridge is about four times as expensive as a reel tape. This combined with the fact that with smaller block sizes, more than one cartridge may be needed, means higher prices. I haven't calculated the prices yet, but herewith you are warned.

Again a new distribution tape. This one was tailor made for the EUUG autumn conference in Dublin. Its name is: EUUGD12 "Dublin conference tape". In the following distribution list, there is a brief description of what's on it.

As always, anyone is invited to make their own tools, games, etc available for publication on an EUUG tape. Please contact me for more details. Don't hesitate, just put the results of many nights of serious programming and hacking in the public domain, and you might even become famous!

This is a list of all the current (October 1987) EUUG distributions. It is a short description of the available tapes. Any changes to the contents of the tapes, as well as announcements of new tapes will be placed in eunet.general and the EUUG Newsletter.

Prices of the tapes are in Dutch guilders (DFl), and do not include VAT-taxes or postage. Note that prices for distributions on cartridge tapes will be at least DFl 100,- higher, and possibly even more, depending on the number of tapes used. Note also that you have to be an EUUG member (or a member of a local UUG) to obtain tapes at list prices. Non members will have to pay an extra DFl 300,- per tape.

EUUGD1 R6: UNIX V7 system, specially made for small DEC PDPs (11/23, 11/34, etc). The Kernel supports the UK terminal driver. V7 source licence minimum.

Price: DFl 120,-

EUUGD2: Early Pascal compiler of the Free University of Amsterdam. V7 source licence minimum.

Price: DFl 120,-

EUUGD3 R3: UNIX Networking software, "news", and some auxiliary programs. A fairly debugged version of UUCP and X.25 support will be added if a copy of the source license agreement for at least UNIX version 7 is included.

Price: DFl 60,-

If requested, two tapes containing the major news-groups received on the Continent for the last several months are available.

Price: DFl 240,-

- EUUGD4: Software tools, sampled by the Software Tools Users Group. Most of the software is written in Ratfor, for which a Fortran support tool is included. This tape is available in different formats: DEC RSX, DEC VMS, UNIVAC, IBM MVS, UNIX tar, MIT line feed format, and MIT card format (80 columns).
Price: DF1 150,-
- EUUGD5: A collection of benchmark programs made up by EUUG.
Price: DF1 60,-
- EUUGD6: (USENIX 83.1)
USENIX tape, containing contributions from various UNIX System Group Members. This is a licence dependent distribution: V7, V32, SIII, V6 or no licence disclosure available.
Price: DF1 240,-
- EUUGD7: UNIX:STAT Version 5.2. A collection of about 25 data manipulation and analysis programs written in C by Gery Perlman.
Price: DF1 60,-

EUUGD8: A collection of useful software, based on the so called Copenhagen tape (EUUG UNIX conference Autumn 1985).

EUUGD9: A collection of useful software, based on the so called Florence tape (EUUG UNIX conference Spring 1986). Price: Dfl 150,-

EUUGD10: MMDFIib. Multichannel Memo Distribution Facility (version Iib). This is a powerful, domain oriented mail system with access control and the ability to communicate over a variety of network systems including TCP/IP, JANET, UUCP, PHONENET, etc. It has been ported to a variety of UNIXs including but not limited to 4.[123]BSD, 2.9BSD, System III/V on a variety of different hardware. You should first obtain a licence agreement by sending a message to euug-tapes@mcvax. Return the signed licence with your order. Price: Dfl 90,-

EUUGD11: This is the "Boat" tape; the Helsinki EUUG 1987 spring conference. It contains about 25 Megabytes of programs, games, etc. Including:

```

gprolog
graph
jove      editor
less
make
nag      an annoying reminder service
news     ``The Universe, some information to help you live it ...''
pdtar    public domain tar
rn       the net-news-reader tool by Larry Wall.
sc       spread sheet calculator
shar     and unshar: the Shell archiver
teco     editor know from DEC systems
uEmacs
uuencode      and uudecode: encrypt the love letters to your
lover ...
vn          visual news
xmodem     a means to transfer programs.
zmodem     yet another way to transfer stuff.
yacpps
larn      a game
pathalias
uumail
stat      version 5.3 of Gary Perlman's statistical package.
gnu bison
gnu chess
gnu compress
gnu emacs
gnu hack      if you don't know this one ...
gnu scheme

```

Price: Dfl 120,-

EUUGD12: This is the Dublin EUUG 1987 Autumn conference tape. It contains about 26 Megabytes of programs, games, etc. Including:

```

copytape
crc_plot

```

cwb
elm
fastgrep
jove
kermit
logo
notes
uupc For Atari, DOS machines, Macintosh.
adl
nethack
cron
ease
mcp
sendmail
uumail
minix Not the complete source, but some enhancements.
mh Version 6.5
Recipes The complete archive. Bon appetit!
brl-gw Gateway code
isode
pcip
pctelnet

Price : Dfl 120,-

1. *EUUG Tape Distributions Order Form*

If you want to order any tape, please write to:

*EUUG Distributions
c/o Frank Kuiper
Centrum voor Wiskunde en Informatica
Kruislaan 413
1098 SJ Amsterdam
The Netherlands*

For information only:

*Tel: +31 20 5924056 (or: +31 20 5929333)
Telex: 12571 mactr nl
Internet: euug-tapes@mcvax (or: frankk@mcvax)*

Please note that for distributions D1, D2, D3 and D4 (and in some cases also for D8) a copy of your source licence agreement with .SM AT&T for at least UNIX Version 7 should be enclosed.

Note also that you have to be an EUUG member (or a member of a national UUG) to obtain tapes at list prices. Non-members will have to pay Hfl 300,- extra per tape as handling fee.

Please enclose a copy of your membership or contribution payment form when ordering.

All tapes come in **tar** format, 1600 bpi. 800 bpi is possible on request. Tapes and bill will be sent separately.

EUUG Tape Distributions Order Form

This page may be photocopied for use.

Name:

Company:

Address:

.....

.....

I would like to order the following:

.....

.....

.....

.....

EUUG (or national UUG) membership form enclosed? Yes / No

Copy of AT&T source license enclosed? Yes / No

"I indemnify the European UNIX systems User Group for any liability concerning the rights to this software, and I accept that EUUG takes no responsibilities concerning the contents and proper function of the software."

Signature

Date

Letter from Germany

Achim Brede
GUUG
achim@bredex.uucp

Bredex GMBH
Braunschweig, W Germany

This is the first "Letter from Germany" since Daniel Karrenberg left the GUUG governing board. I would like to use this letter to thank Daniel for all the work that he has done for the GUUG and the EUUG. As a tribute to him I shall continue using the same title for these articles.

Since April 1987 I have been the vice chairman of the GUUG. This position is tied to the function of the EUUG contact person. I took over this position over at a time when the GUUG had financial problems because of the EUUG subscription, which we consider to be high. At that time it looked as though the EUUG was not willing to discuss topics like fees or services and their costs, but at the governing board meeting in Helsinki the idea was born to organise a workshop on the future of the EUUG. This meeting would result in some kind of brainstorming, where all the national groups of the EUUG would try to define new (or old) goals to fulfill the actual and future needs of the EUUG and their national groups.

The EUUG now celebrates their 10th anniversary. Congratulations!!! Many things have changed in 10 years. Starting as a group with direct members in some countries, there are now 14 countries running their own groups. Maybe there are not only more UNIX users in Europe; it could be that today's profile of the UNIX user has changed. I will say no more on this topic and you should refer to the report of the Paris woods meeting.

A few weeks ago we had our 1987 GUUG conference. It took place in Karlsruhe on the 22-24th of September which was unfortunately the same date as the Dublin EUUG conference. I have to apologise for that, but it was impossible to find a different location or a different date for an event of that size. The conference had four parallel sessions, two technical in large lecture halls and two commercial sessions in smaller rooms. I do not have the exact numbers but I suppose there were more attendees than last year. There was an surprisingly high interest in the *ad hoc* working groups "network" and "user interfaces".

Finally I will give you the dates of the next events in Germany.

Berlin, March 24-25, 1988

GUUG Workshop

Working groups: "networks"; "user interfaces", and "security"

Only a limited number of attendees are allowed per working group. The groups are working in parallel.

Hanover, September 27-29, 1988

1988 GUUG annual conference.

This conference will be held in conjunction with the Autumn 1988 EUUG conference.



News from Denmark

Keld Simonsen
keld@diku.dk

Chairman, DKUUG

So what has happened in Denmark since last time we met here? Quite a lot.

Last time I wrote about the wonderful machine that we were given to run the network. Well, actually we did not get it as our own, we just borrowed it: for one year. But that does have some advantages, because the firm is doing all the maintenance on their expense, and it is easier to shift to another machine if this machine (a SUN 2/160) gets too small.

We have not yet moved the backbone to this machine, due to many causes like too little disk space, summer holidays, lack of manpower, disk troubles and what have you. Ericsson A/S has promised us another disk, so we'll have 260 Mb in total.

And there has been other problems with the network. Nothing but trouble... The institute of computer science at University of Copenhagen (DIKU) announced in April that it would stop support for hooking up new machines to the network, and that DKUUG — the Danish EUUG branch — would have to do that work. It was our network anyway, they said, and they did not really have too much interest in contacts to all these commercial firms using the network. But, they were willing to offer continued systems support and machine power for the network. Well, it is understandable when people stop doing work for free, and DKUUG is pleased that there is still a machine and some support to run the network.

The DKUUG board then decided that the network activities were to run economically, self contained and without loss, so we had to raise the prices by about 50% to make it possible to hire someone to do the contacts to new and old net users. We only lost a couple of users by that move and more than doubled our news subscriptions, from 5 to 11 sites, with about 45 machines on mail. Half the new salary is paid by fixed subscription fees, and the rest is paid by a flat volume charge per Mb. We then cover our telephone and X.25 transmission costs (both national and international) by billing them directly to our users. The news cost from Amsterdam is split evenly on the subscribers.

Another formal thing about the network is that DKUUG is negotiating a written agreement with the Computer Science institute, thus we will know what rules are applying to this cooperative effort. It was no great pleasure to just get the message that vital parts of the net service would stop about a week ahead. Too many people are now relying on the UNIX network.

The moral I got out of this story is that is it fine if some institute or company will run the network in a country (and take all the work, troubles and financial risks with it), but you can never tell when they will stop providing those services or parts of it. As it is UNIX users using the network, from both the commercial and academic worlds, it shows up that the only organisation really interested in keeping the network running is the national UNIX user group. They will take all the responsibility if things fail fatally. So they should better be prepared to prevent the network from failing. I am not advocating that the national UNIX group should actually run the network, but some formal arrangements with the backbones would

give some security.

Well, as it showed up in the arrangements with DIKU, they really wanted to shed the financial responsibility of the network, so DKUUG had to take on this job too. And though other sites were willing to take over the backbone role, no one was interested in taking the financial risks. So DKUUG has to take that responsibility from now on, whether we like it or not.

Enough on the network.

We have also had 3 members meetings, on 4th generation languages, UNIX introduction, and work-stations; with 80, 50 and 50 attendees respectively. We think that we are quite successful in having good themes for the meetings and aiming them at different audiences in our membership. In this way there is something for users at all level. We also find that written material with extensive abstracts is very important when we announce the meetings. This is the information the members decide upon, if they want to attend.

We have a special working group with our members who are UNIX products suppliers where we discuss exhibitions, catalogues and the like. We are planning a DKUUG stand at the Mikro Data 88 exhibition in Copenhagen, February 1988. We are also planning a list with indexes many of the UNIX products available in Denmark, but without a wordy description of every product, only keywords will be allowed. People would then refer to the /usr/group catalogue, or just call the supplier and ask for details.

Yes, DKUUG is in the process of affiliating /usr/group, and we intend to use their services, like the catalogue, too.

Other external relations: we have meetings with the local X/OPEN consortium and we are active in the official ISO standardisation efforts on C and UNIX (POSIX) through membership of the appropriate subcommittees in the Danish Standards Association.

A last comment: there are now (1st November 1987) about 220 members in DKUUG. This is quite far from the goal stated in our last report here, but we find it quite good compared to the size of our country.

EUUGN Dublin abstracts

Here are brief abstracts of the papers that were presented at the Dublin conference on September 23rd — 25th.

Copies of the proceedings are available from the EUUG secretariat at Owles Hall. The cost is £20 for members of the EUUG, and £50 for non members.

A Framework for Man Machine Interfaces Design
Laboratoire de Recherche en Informatique, France
Michel Beaudouin-Lafon and Solange Karsenty
mbl@sun1.lri.fr so@lri.lri.fr

With the increasing importance of man-machine interfaces appears the lack of tools for the construction of user interfaces. Some principles are coming out, such as the separation of the interface from the application, and allow us to start designing environments for their construction. To facilitate their construction means to design fast, to modify easily and eventually to reuse. On the user side, the interface should be dynamically adaptable to his taste and experience.

Towards such interfaces, we propose an environment built upon two tools: Graffiti and MetaGraph. The first one allows for interactive construction of the control structure of the interface, while the second one establishes a relation between the program data and the interface. This correspondence of high level abstractions will extend the functionalities of the interface by reducing graphic manipulations in the application.

What They Don't Tell You about Window Interfaces
Michael D. O'Dell
Maxim Technologies Inc, USA
mo@maximo.uucp

People communicate using languages, whether it be with other people or computers. The nature of these languages determines what can be expressed and how effectively it may be communicated. These languages may be in several forms, verbal and visual being just two of them. How should the design of a computer interface language be set about, and then how do you persuade the different systems designers to stick to one of them?

MUSK — a Multi-User Sketch Program
Chris Crampton
Rutherford Appleton Laboratory, England
cmc@vd.rl.ac.uk

In recent times we have witnessed an explosion in the use of personal workstations, but the development of interactive applications has lagged behind the capabilities of the hardware. This is particularly evident in the area of user-to-user communication.

The subject of this paper is a highly interactive sketch program that allows any number of users to participate in the sketching activity. In addition to sketching, facilities are provided for the inclusion of text and the pasting of images.

The paper starts with a brief retrospective of the user-to-user communication capabilities of UNIX and progresses to describe the design and implementation of a multi-user sketch pad.

The paper concludes that such a facility is both feasible and useful although a number of questions remain about how multi-user and media-user conversations should be managed.

UBOAT — A UNIX Based On-line Aid to Tutorials

Daniel V. Klejn

Carnegie Mellon University, USA

dvk@sei.cmu.edu

Designers of computer aided education (CAE) systems are presented with the dilemma that course authors are required to also be expert in the field of computers. While this is a laudable virtue, it is very often the case that a skilled educator is not necessarily a skilled programmer and, conversely, a well trained programmer is rarely a good educator. Consequently, CAE packages are either rich in information but difficult to use, or else they are wonderful examples of human-machine interfaces which convey very little real information to the student. This paper describes our work in developing UBOAT, a system designed to allow an educator with little computer expertise to develop an interactive, dynamic and effective CAE system. UBOAT allows an educator to concentrate on his or her area of expertise, without requiring a great deal of distraction in coercing the computer to interface effectively with the learner.

Early attempts at generalised CAE generators such as *Learn* or *PILOT* provided an educator with simple interface to the student that allowed a predominantly top-down approach to presenting the educator's materials. Little or no attention was paid to allowing for review, question and answer, ergonomic human-machine interface, or screen oriented display. *Learn* presented all of its text in serial form, and generally required modifying the source program to add features. *Pilot* allowed for simple graphics, but required that it be done in a terminal-dependent way. Both these systems, as well as many others, had a cryptic, terse command syntax that was fairly unforgiving. The source files for the education scripts were often huge ASCII files, without any data compression or text reuse and rarely, if ever, was any debugging support or compilation facilities provided.

Developing Ada Software Using VDM in an Object-Orientated Framework*

Chris Chedghey, Seamus Kearney, Hans-Juergen Kugler

Generics (Software) Limited, Dublin, Ireland.

chedghey@genrix.uucp

A software development method encompassing a variant of Object-Orientated Design (OOD) and the Vienna Development Method (VDM) is described. The development is targeted at the Ada programming language. Part of the objective of the work described here is to produce a method for constructing Ada software — a method which follows sound engineering principles in order to improve software quality as well as improving productivity on the part of the developers.

Papillon — Support Tools for the Development of Graphical Software

Chris Chedghey

Generics (Software) Limited, Dublin, Ireland.

chedghey@genrix.uucp

The Papillon project (ESPRIT 496), entitled "A configurable graphics subsystem for CIM", has produced a prototype system to help the application developer build well-structured software which is initially devoid of representation or user-interface information. Graphical representations for application data types may be subsequently described without direct coding. Calls made to the application by "driver" software such as process control or simulation programs will now be reflected in the display. A user interface may be

* *Ada is a registered trademark of the US Department of Defense (AJPO)*

EUUGN DUBLIN ABSTRACTS

automatically generated to enable a human user to "drive" such applications as production planning and scheduling. The approach has its foundations on Object-Orientated Design (OOD), the Vienna Development Method (VDM) and the programming language Ada.

Experiments With the User Interface of UNIX Mail

*Peter R Innocent, Gerrit C van der Veer, Yvonne Waern
Leicester Polytechnic, England & elsewhere*

Designing the user interface and application interface are only two aspects of the complex work that has to be done to provide users with systems that they both are able to use, and like to use. An interdisciplinary approach is necessary to define, design, and construct these interfaces. For the COST-11-ter working group "Human Factors in Telematic Systems" electronic mail was used as an example to illustrate the way to develop user interfaces. As we wanted to start from an existing situation (an application system in actual use), we chose an application that is certainly neither the best, nor the most advanced. It is, however, a standard component of UNIX which is becoming a standard operating system for a growing diversity of generally available machines.

A SunView User-Interface for Authoring and Accessing a Medical Knowledge Base

*Neil P Goundwater
Sun Microsystems, Inc., USA
npg@sundc@sun.com*

A system named *Eidetic* has been built for the storage, retrieval, and examination of microscope slides. It is made up of two major subsystems: an authoring system, and a searching system. After an author collects images and composes a knowledge base both get transferred to Compact Disc Read Only Memory (CD-ROM). Searchers can analyse the stored data to draw conclusions and assist their diagnostic process. *Eidetic* supplies "power tools" to the physician or medical student.

The paper covers the overall design of the system to highlight the components which are affected by *SunView* programming techniques. To acquaint the reader with the goal of the delivered system, sections will also cover the world-view of the users, an overview of the knowledge base and its use, describe the authoring and user systems, and outline the organisation of the CD-ROM storage.

The Analysis and Manipulation of BNF Definitions

*Allan C. Milne
Dundee College of Technology, Scotland*

An extended BNF notation (EBNF) is formally defined and its use described. An association software tool called LL1 provides analysis and manipulation functions for systems defined using this notation. The facilities provided by LL1 are particularly useful in the development of LL(1) definitions. This paper will also discuss the use of EBNF and LL1 in the development of language syntax and will consider some implications of using EBNF representations of Jackson structure diagrams.

UNCLE — A Case Study in Constructing Tools for the PCTE

*Hans-Juergen Kugler, Barry Lynch
Generics (Software) Limited, Ireland.
lynch@genrix.uucp*

A set of common service interfaces has been proposed as basis for a portable common tool environment (PCTE). The PCTE favours a distributed workstation approach to software engineering environments and uses an object management system (OMS) based on an entity-relationship approach as fundamental information repository.

The PCTE defines functional interfaces allowing the creation and manipulation of complex data-models using the distributed OMS, thus facilitating the interpretation of development and

management tools. The PCTE interfaces are currently based on UNIX services and expressed in C.

UNCLE is a tool to allow the construction of command and response languages for the PCTE which are turned to the users' views of the underlying system and his or her skills. An object-orientated abstraction mechanism allows the construction of common tool interfaces.

A Distributed Design Environment for Distributed Realtime Systems

Christoph Senft

Technical University of Vienna, Austria.

senft@vmars.uucp

Over the past few years the need for systematic design of real-time systems has received considerable attention due to the increasing criticality and complexity of the applications. This paper presents a distributed project support environment for the design of fault-tolerant distributed real-time systems. The innovative aspects of this environment relate to the handling of real-time from the requirements phase to the detailed implementation stage and the integration of the design tools with the evaluation tools. Data refinement, function refinement, evaluation and project management are supported by an open, coherent tool set. The uniform man-machine interface is one of the integrating factors of the whole environment.

The design environment is implemented under the UNIX operating system on a state-of-the-art graphic workstation with pointing devices. The X-window system supported the creation of the window and icon driven user interface. All the information gained and used during the design process is stored in an entity-relationship model implemented by the relational database DB++.

NERECO: An Environment for the Development of Distributed Software

Giandomenico Spezzano, Domenico Talia

CRAI, Italy, and Università di Pisa, Italy

dot@crai@12unix

NERECO (NETwork REmote COmmunications), the system here described, is constituted by a set of tools for the development and execution of distributed programs. From an architectural point of view the tools supporting such programs have been realised through the enrichment of many sequential languages (Pascal, C, CHILL) with concurrent statements and the run-time support for such statements. The integration of the concurrent part into the sequential languages has been done in order to preserve the syntactic coherence.

The co-operation model utilised is derived from CSP, with some extensions to allow asymmetrical and asynchronous communications; moreover, some statements for fault treatment have been added. The run-time support implements the synchronisation and the communication among the processes; for its implementation the IPC of UNIX 4.2 BSD, and particularly the TCP/IP protocol, has been utilised. The NERECO system has been implemented in the C language on a collection of SUN workstations connected by Ethernet.

A Knowledge Based CAD System in Architecture on UNIX

S. Hanrot, P. Quintrand, J. Zoller

E. Chouraqui, P. Dugerdil, P. Framcois, M. Ricard

GRTC/CNRS, Marseille, France.

crtc@frmop11.bitnet

The aim of this project is to realise an intelligent CAD system integrating part of the Architectural knowledge. It is mainly based on the assumption that this knowledge is partially included in the vocabulary of the architect and in the drawing techniques elaborated along the centuries.

In this system the symbolic representation of knowledge is based on an object-oriented language, OBJLOG, which we purposely defined as a layer above PROLOG II, and which offers some interesting processes of inheritance.

EUUGN DUBLIN ABSTRACTS

To implement the geometric model we have chosen SMALLTALK-80, an object-oriented language provided with a rich graphic programming environment, which permits us to realise a quite sophisticated user interface (pop-up menus, mouse device controls...). Communication between the model knowledge and the geometric one is carried out by UNIX in running Objlog and SmallTalk as two concurrent processes.

A User Interface for Geographers — What Can UNIX Offer ?

Roger Bivand

Nordland College, Morkved, Norway.

Geographers have not traditionally used computers much. This paper describes the problems peculiar to this discipline, largely modeling and statistics, both on a grand scale. Emphasis is placed on picking out key features from a mass of noise. Computers are now beginning to be used to great effect, but finding the correct tools has been difficult.

The HUB: A Lightweight Object Substrate

Michael D O'Dell

Maxim Technologies Inc, USA

mo@maximo.uucp

This paper describes the Hub system which is a machine emulator capable of operating under many different contexts in quick rotation. Tasks running in Hub pop their next instruction into a circular buffer. Creating a new task is largely a matter on popping two instructions, one of which has a different context. This is considerably more efficient than rapid context switching, which can be quite slow on some machines. Hub was initially designed to do communications processing, an area where many tasks execute and communicate concurrently.

More MIDI Software for UNIX

Michael Hawley

MIT Media Lab, Cambridge, USA

mike@media-lab.mit.edu

The Media Lab has a Sun-3 which controls 64 channels of Midi synthesisers in real time. In addition to capturing and replaying music, work has been done on analysing it. An early example is one which will tell you which key a piece is played in. Other analyses look at note frequency and rhythmic patterns, and will eventually lead to recognition of a composer's style.

Automatic Exploitation of Concurrency in C: Is It Really So Hard?

Lori S. Grob

Courant Institute of Mathematical Sciences.

grob@cmcl2.nyu.edu

In this paper we consider whether it is possible to have compilers that will automatically exploit concurrency in C. We discuss the relationship between automatic exploitation of concurrency for the purposes of vectorising and multiprocessing. We review the basic techniques and transformations used and examine the necessary conditions to perform these transformations, with examples in C. Several elements of the C language and programming style, such as pointers and recursion, make it difficult to do the necessary data flow analysis. There are two possible approaches to this problem: to bypass code or blocks of code that contain "difficult" features and be unable to apply optimisations to these fragments, or to suitably restrict the language. We examine the choices made by the few available vectorising and parallelising C compilers and consider what the future may hold in the light of current research.

An Intelligent, Window Based Interface to UNIX
 Stuart Borwick, John R.Nicol, Gordon S.Blair
 University of Lancaster, United Kingdom
 stuart@dcl-cs.uucp

Many operating systems have been developed with little thought given to the user interface. Most interfaces tend to be awkward to use and rely on simple command line interpreters. Although many experienced users have little trouble with the UNIX interface, the popularity of the system demands that a more user oriented interface will be required in the future. One aim of this interface should be to alleviate the need for naive users to learn and remember a large number of command names and cryptic command lines.

The UNIX interface can be improved by employing the facilities offered by bit-mapped graphical workstations (menus, windows, icons, etc.). To use these facilities most effectively, the interface requires intelligence. This can be provided by associating semantic information with UNIX OBJECTS. This paper describes a UNIX interface which uses a mode of interaction known as *direct manipulation*. Direct manipulation assumes an environment of objects and associated semantic information (i.e. attributes). It encompasses the ability to select objects with a pointing device, display and issue commands associated with the objects and change attributes of the objects.

This kind of interface helps a user in several ways. Displaying UNIX objects (files, directories, etc.) as icons provides a more natural interface. In addition, the association of semantic information with objects makes it easier to provide a more intelligent user interface. For example, with typing information the system can deduce exactly which commands are permissible in a given context and does not rely on commands themselves to detect errors. Consistent and helpful error messages can also be generated. Finally, the use of direct manipulation, combined with semantic information, permits a much reduced command set to be displayed in a given user context. The authors see this as an effective way of assisting the user.

Fast bitblt() with asm() and cpp
 Bart N Locantli
 AT&T Bell Laboratories, New Jersey, USA

bitblt() is both a fundamental building block of bitmap graphics and a demanding implementation challenge. bitblt()'s speed, or lack thereof, is an important factor affecting one's perception of a machine's interactive response.

This paper is about squeezing the last microsecond out of software implementations of bitblt(). The environment for this exercise is a bitmap display based on the Motorola 68020. Both the asm() capability of the compiler and the C preprocessor will figure prominently.

DES — Support for the Graphical Design of Software
 Stephen Beer, Ray Welland, Ian Sommerville
 University of Strathclyde, Scotland, and University of Lancaster, England.

This paper describes DES (the Design Editing System) — a system which investigates how such graphical support may be provided in a *generalised* way for software engineering purposes. DES comprises of 3 tools: a shapes editor (SHAPES) for defining the shapes of a method, a language (GDL) for describing a software design method and a graphical design editor (DE) which is driven by tables generated from the first two tools. The system is implemented in C on a Sun workstation using the pixrect graphics layer and the panel user interface package.

At a very high level of abstraction a design diagram can be viewed as a number of symbols and a number of rules concerning the physical and logical constraints on those symbols. A novel feature of DES is that it is not geared towards any specific method. Rather, the tool builder defines the syntax, semantics and shapes of a design method using the high level tools SHAPES and GDL. This increases the applicability of the system since MASCOT and JSD users, for example, may utilise the same system facilities.

EUUGN DUBLIN ABSTRACTS

This paper presents each of the tools, emphasising how method-specific checks may be specified in GDL and enforced during design editing sessions.

Cleaning Up UNIX Source — or — Bringing Discipline to Anarchy
David Tilbrook, Zalman Stern
Carnegie Mellon University, USA
dt@andrew.cmu.edu, zs01@andrew.cmu.edu

Many people are coming to the realisation that the distribution of UNIX software is not just a matter of creating a tar tape. Approaches to software and distribution management vary from simple disciplines, conventions and procedures to full scale Integrated Programming Support Environments (IPSEs). This paper discussed some of the problems and issues involved in the management of software, concentrating primarily on the exporting of source to remote locations.

EUUG

European UNIX[®] systems User Group

SPRING 1988 Conference
“UNIX[®] around the World”

London 11th – 15th April 1988



For further details, contact: The Secretariat **European UNIX[®] systems User Group**

Glossary

This newsletter contains about 5400 different words, some simple, some not so. Below is a list of the words that may be difficult for people for whom English is not their first language. The words below are explained in the context in which they are used in the newsletter. There are other meanings to some of the words.

advocate	recommend, support
aspect	way of looking, component
avant-garde	pioneers, forward looking
brainstorming	spontaneous discussion in search for new ideas
buck	dollar, money
bugbear	persistent problem
coerce	make obey, persuade
coherent	self consistent
compliance	act in accordance with
consortium	association of several people/companies
conspiracy	secret plans for an unlawful purpose
convey	impart/communicate an idea
convince	persuade to believe firmly
cranked	produced, made
dearth	scarcity, not many of
deduce	infer, draw as logical conclusion
deficiency	thing lacking, missing feature
despite	in spite of, something happening regardless of attempts by something to stop it
devoid	without, quite lacking
eclectic	selected so as to please, of several sources
embark	start a project
encompassing	surround, contain
enigma	puzzle
ergonomic	efficiency of people in a working environment
extrovert	person of an outgoing personality
feasible	(something) technically possible to do
hype	over-enthusiastic advertising/marketing
indemnify	legally protect against consequence
innovative	producing new ideas, changing products
killjoy	person who stops enjoyment
lagged	followed (in time) behind ...
liability	legally responsible for ...
perception	viewpoint, how one sees something
prescient	seeing what is to come
prosaic	unromantic, lacking poetic beauty
prudent	wise caution/action against ...
relate	tell
relate	have reference to, be connected with (as in family)

royalty	payment to (eg) an author of a book
snag	(unforseen) problem
superlative	of highest degree or quality
tame	make domesticated/gentle
terse	brief, short
unique	only one in existence of ...
weird	very strange/unusual
yield	give up a secret, give result/product

*UKUUG News Item**Sunil K Das**sunil@cs.ucl.ac.uk**The City University**London*

UKUUG are holding a UKNET Workshop on Monday 14th (pm) and Tuesday 15th (am) December in the Oliver Thompson Lecture Theatre, City University, London. The event will begin with lunch at 12.30 on Monday. The Workshop is being held between 14.00 and 17.30 on Monday, and 09.30 and 12.30 on Tuesday, followed by lunch before departure. An evening meal and bar will be provided. Accommodation in one of the University's Hall of Residence will be bookable.

The UK backbone is sited at the University of Kent at Canterbury under the direction of Peter Collinson. The workshop will be chaired by the UKUUG chairman Sunil Das with Peter coordinating and directing the technical presentations.

Ten of the UK's leading experts on various aspects of networking have volunteered to address the audience about intermediate and advanced technical topics; the former being discussed on Monday followed by a Question and Answer Session and the latter being discussed on Tuesday. As a consequence, the booking arrangements will allow both half days or one of the half days.

Full booking details may be obtained from the UKUUG secretariat by telephoning Bill Barrett on 0763 73039 or sending email to him at ukuug@ukc.ac.uk. Any other enquiries can be sent to the UKUUG Chairman sunil@cs.ucl.ac.uk.

EUUG

European UNIX® systems User Group

Are YOU a UNIX® User?

If so, you can dramatically extend your knowledge of UNIX systems and share your views and expertise with other UNIX users throughout the world by becoming a Member of the

European UNIX systems User Group

The EUUG is a non-commercial organisation which is in-being to:

- * Enable regular contact between UNIX users
- * Provide a source of unbiased information
- * Provide forums for discussion by means of exhibitions, conferences and symposia
- * Improve knowledge of UNIX and thus improve the System itself

In addition to a quarterly **Newsletter**, the EUUG makes available its EUNet **electronic mail** and news services to all Members; supports the activities of its National User Groups throughout Europe; provides a **Software Distribution Service**; and has a large number of specialists within its Membership to help users with their problems.

Its International Conferences are now a very important part of the UNIX scene with future events scheduled to take place in Finland and Sweden (May 1987); Dublin (Autumn 1987); and London (Spring 1988).

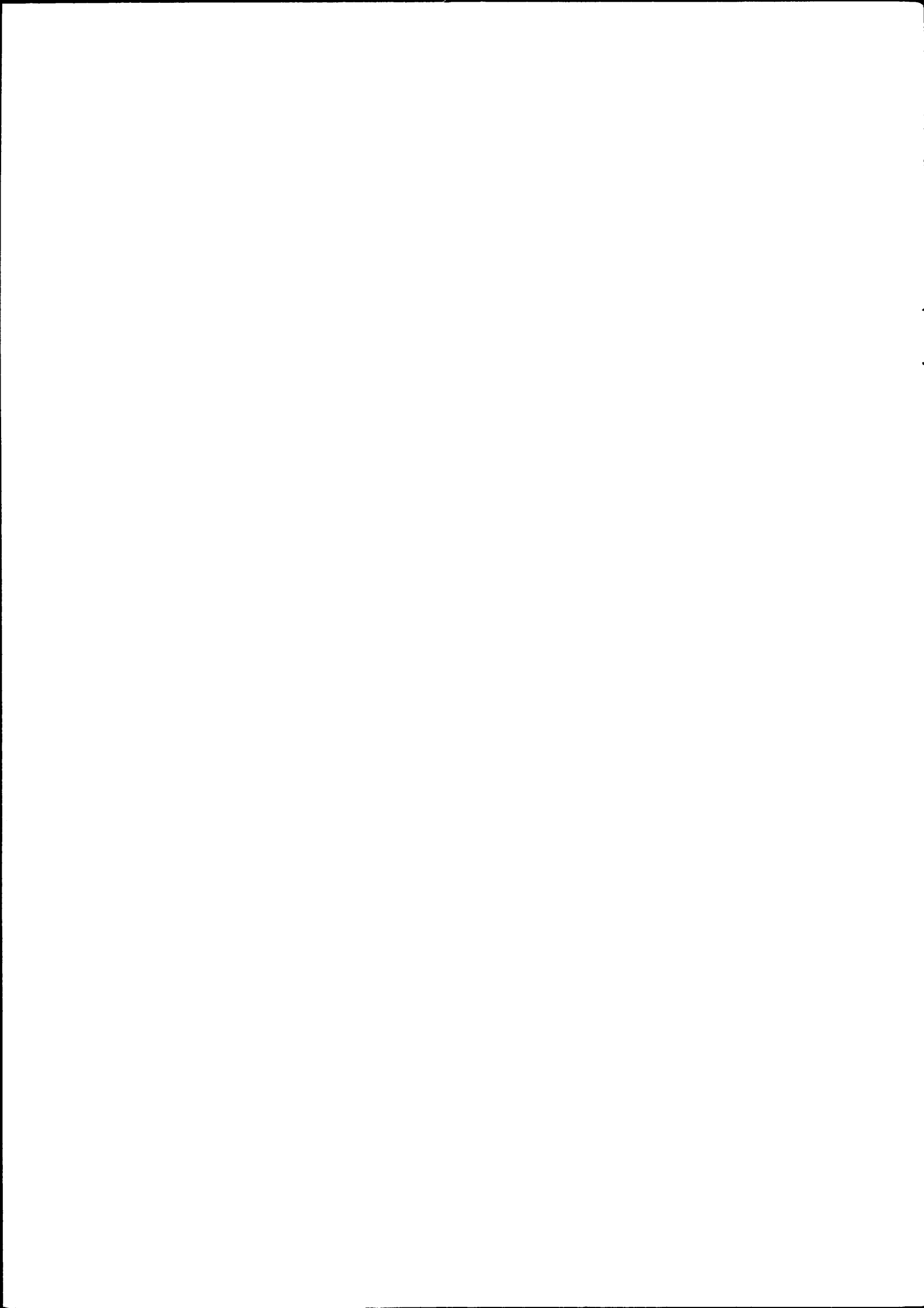
If you want to play a bigger part in promoting the cause of the UNIX Operating System and — at the same time — benefit from the many services offered by the EUUG, then contact the Secretariat NOW for further details.

European UNIX® systems User Group

Owles Hall, Buntingford, Herts SG9 9PL, England Tel: 0763 73039 Network address: euug@inset.uucp

UNBIASED INFORMATION,
ELECTRONIC MAIL,
SOFTWARE DISTRIBUTION,
INTERNATIONAL CONFERENCES





The Secretary
European UNIX[®] systems User Group
Owles Hall
Buntingford
Herts. SG9 9PL.
Tel: Royston (0763) 73039.

*UNIX is a Registered Trade Mark of AT&T in the USA and other Countries