NEWSLETTER

EUUG

European UNIX® systems User Group

Volume 8, No. 2
Summer 1988

CONTENTS

EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

EUUG

Volume 8, Number 2
Summer 1988

Editorial Team:

Philip Peake	Axis Digital, France (EUUG Newsletter Editor)
Alain Williams	Parliament Hill Computers ltd.

Typesetting:

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

Printed by:	Rank Xerox (Copy Bureaux) Ltd. 68-74 Rochester Place London NW1 9JX U.K.
Published by:	The EUUG Owles Hall, Buntingford Hertfordshire SG9 9PL U.K. euug@inset.co.uk (... !mcvax!ukc!inset!euug)

ISSN 1011-4211

Editorial

Alain Williams
addw@phcomp.co.uk

Parliament Hill Computers Ltd
London NW3 2TS
UK
+44 1 435 0200

There are more and more of you

As the membership of the EUUG grows so do the activities of its members. Many of you were at the very successful London conference — the biggest European Unix conference to date.

The membership of the national groups is (February figures):

DKUUG	216	UKUUG	229
FUUG	154	IUUG	31
EUUG-S	286	UNIGS	18
AFUU	586	GUUG	324
i2u	100	UUGA	69
NLUUG	212	BUUG	138
ICEUUG	21	NUUG	97

Future Issues

I am interested in receiving papers for publication from readers. Although the official language of the Newsletter in English I am happy to receive papers in any European language.

Articles should be sent to me at the above address. The idea form is by e-mail using the macros.

I can also accept articles on the following floppies: 3b2 5½", IBM-PC 5½" and 3¼" high and low densities, Apple Mac 3¼".

I have a template for article layout — please mail me for a copy.

The next copy dates are:

25 July	for	1 September
24 October	for	1 December

Advertising in the EUUG Newsletter

The EUUGN is now accepting publicity for inclusion in its pages. This is subject to the following conditions:

— The cover (front, back and inside, out) will not normally be available.

— There is limited space available. Space will be allocated on a first-come first-served basis with preference being given to EUUG members.

— Material should be provided in either "camera ready" or "troff source" format. The latter case should be accompanied by a paper copy.

Colour may be possible — more details from the EUUG secretariat.

— The EUUG reserves the right to refuse any advertisement.

The current pricing scheme is as follows:

— EUUG Members
  — Full page — 300 GBP
  — Half page — 175 GBP
— Non Members
  — Full page — 450 GBP
  — Half page — 250 GBP

The deadline for receipt of advertising material is the same as that for articles, the actual dates can be found in preceding issues of the EUUGN.

It is not possible to do country specific advertising as part of the EUUGN, but country specific inserts (into the envelope) can be made.

For full details please contact the EUUG secretariat.

Extra Copies

Institutional members of the EUUG may now receive more than one copy of the EUUGN. The price will be just sufficient to cover the extra production and handling costs. All copies will be sent to one address.

Please contact your National group for details.

Newsletter Layout

I wish to welcome Laura Dekker who has taken over from Sally. Laura started out as a chemist and after inhaling a few too many organic solvents decided that PostScript was safer as well as being much more fun.

In this issue of the newsletter we to have moved to 2-column format, and made a few other changes in its general appearance. We hope that this will make it easier to read.

The headings are all unnumbered except where the author has especially requested otherwise. You'll also notice the EUUG logo at the top of each page, and one or two other items — courtesy of PostScript!

Happy Events

Congratulations are due to Rik and Teus Hagen on the birth of the baby girl. Femke weighed in at 2.8kg at about the time of the London conference. They are both reported to be 'over the moon' at the birth of their first child.

Neil Todd has announced his engagement to Mary Ann Doig. We wish them every happiness in their future life together.

Jill and I had a baby boy, we are calling him George. He weighed in at 8lb 3oz, but being British, he doesn't understand Kg!

A letter from the Chairman

In the winter 1988 edition of the EUUG Newsletter an EUUG executive board article was published. This article was originally drafted by Keld Simonsen; at that time a member of the EUUG executive board. Due to time constraints modifications were made to the structure and contents of this article and the article printed without consultation with Mr Simonsen.

Apologies are offered that the resulting article does not correspond fully with the original ideas of the author and that it was published under his name. The responsibility for this article is fully that of the executive board. Actions have been taken to ensure that such events will not occur in future.

Teus Hagen
Chairman, EUUG Executive

The JUNET Environment

Jun Murai
jun@utokyo-relay.csnet

University of Tokyo
2-11-16, Yayoi, Bunkyo
Tokyo, 113 JAPAN

JUNET has been developed in order to provide a testing environment for studies of computer networking and distributed processing by connecting a large number of computers and by providing actual services for users. The environment provided by the network represents the special requirements and problems of Japanese UNIX environment in general. Throughout the development stages of the JUNET environment, mechanisms to manage resource naming, Japanese character handling, and fast dial-up link using IP protocol have been developed for the network.

In this paper[1], the current status of JUNET focusing on the special environments and technologies to provide them are introduced.

Introduction

JUNET[1] has been developed as the first attempt to establish an electronic communication environment in Japanese research and development communities. Text message exchanges such as electronic mails and electronic news have been provided as well as other advanced network services. Several international links to world academic networks have also been established. Since this is the only working environment for experimental studies on computer networking in Japan, various research topics including physical communication technologies, network protocols, network interconnections and distributed processing environments are in progress.

The purpose in the very first stage of the network, thus, was to provide an actual network services to researchers and put Japanese communities into worldwide academic networks. And then, we started to work to solve problems existing on the network such as Japanese character handling, name handling for distributed resources, and communication technologies.

JUNET started its operation in October 1984 connecting local area networks in major Japanese universities in Tokyo area[2] and it provides users with the means of the worldwide communications via various international links[3].

The domain addressing over UUCPNET[4] was introduced on May 1985 with a system to generate the address conversion sendmail[5] rules. High speed dial-up modes have been studied in order to increase the transmission rate of the communication. To achieve this purpose, UUCP enhancement with kernel driver development were done, and the efficiency using a dial-up line increased up to more than 13Kbps. This encouraged us to migrate to TCP/IP protocol suite[6][7] even on dial-up lines as well as leased lines. As the result of the development, the dial-up IP link is providing as high as 8Kbps in end-to-end transmission. With general IP transmission over high-speed leased lines and over X.25 public packet switching network, this variety of links introduced us with immediate needs of advanced routing mechanisms which is one of our major field of studies at this point.

As for the internetworking between JUNET and other academic networks is concerned, two gateways are operating to exchanging electronic mails and news. One is Kokusai Denshin Denwa Co. Ltd. (KDD), an international telephone and telegraph company which serves a gateway between UUCPNET/USENET and JUNET, and University of Tokyo is providing a gateway function between CSNET[8] and JUNET which is the major path for most of the other academic networks in the world.

1. This paper was delivered at the EUUG Spring conference but was not submitted in time to be printed in the proceedings.

Generally, there are strong demands for Japanese character handling on computing environment and thus support of Japanese characters by means of computer communications are one of the primary characteristics of JUNET. In order to provide the functions, a JUNET standard *Kanji* code was chosen and conversions between the network standard *Kanji* code and operating system *Kanji* codes are provided by the network application available for JUNET. The general computing environment for Japanese character handling were established as well in order to cooperate with the network environment. The statistics introduced in this paper show drastic influences of Japanese character handling in computer network environment in Japan.

Profile of JUNET

JUNET currently connects more than 1300 hosts in 130 organisations. The geographic areas covered by the network expand from Hokkaido, the northern island, to Kyushu, the southern island, however, concentrations in Tokyo and Osaka areas are obvious as shown in Figure 1. Most of the links have been dial-up lines using 1200 bps or 2400bps modems, although special mechanisms have developed for the UNIX operating system and its communication software UUCP in order to use high speed dial-up modems with 9600bps or higher transmission rate. These mechanisms are also effective for TCP/IP protocols over dial-up telephone lines.

Organisations connecting to the network are Universities as listed in Table 1, as well as research laboratories of computer software/hardware companies, and research laboratories of telephone companies whose domain names are listed in Table 2. All the functions to operate the network have been administrated by administrators at each of the institutes in totally volunteer basis. Table 1: Second level domains for universities in JUNET (Apr. 1988)

Size of JUNET

The size of JUNET can be examined by various statistics. Among them, the constant growth in the number of organisations on the network is remarkable as shown in Figure 2.

One or two new organisations are being connected to JUNET every week on average.

News articles are posted to the network constantly in the `fj` news groups which are currently distributed only within Japan[2] The number of articles posted has been growing as in Figure 3, and 1807 articles are posted in October 1987, as the latest example. Note that only about 15 percent of the articles are written in English alphabet including articles in Romaji, an

Table 1: Second Level Domains for Universities in JUNET (Apr 1988)

`aoyama`	Aoyama Gakuin Univ.
`chuo-u`	Chuo Univ.
`fit`	Fukuoka Inst. of Tech.
`fukuoka-u`	Fukuoka Univ.
`gunma-u`	Gunma Univ.
`hokudai`	Hokkaido Univ.
`kansai-u`	Kansai Univ.
`keio`	Keio Univ.
`kit`	Kyoto Inst. of Tech.
`kobe-u`	Kobe Univ.
`konan-u`	Konan Univ.
`kyoto-su`	Kyoto Sangyou Univ.
`kyoto-u`	
`kyushu-u`	Kyushu Univ.
`kyutech`	Kyushu Inst. of Tech.
`nagano`	Nagano Univ.
`nagoya-u`	Nagoya Univ.
`oita-u`	Oita Univ.
`osaka-u`	Osaka Univ.
`osakac`	Osaka Elec. & Comm. Univ.
`seikei`	Seikei Univ.
`sheart`	Univ. of the Sacred Heart
`shinshu-u`	Shinshu Univ.
`shizujoka`	Shizukoka Univ.
`sophia`	Sophia Univ.
`titech`	Tokyo Insti. of Tech.
`tohoku`	Tohoku Univ.
`tohoku-u`	Tohoku Univ. (Computer Center)
`tokuyama`	Tokuyama National Technical College
`toyo`	Toyo Univ.
`toyota-ti`	Toyota Technological Inst.
`tsuda`	Tsuda College
`tsukuba`	Tsukuba Univ.
`tuat`	Tokyo Univ. of Agriculture & Tech.
`tut`	Toyohashi Univ. of Tech.
`u-tokyo`	Univ. of Tokyo
`uec`	Univ. of Electro-Comm.
`ulis`	Univ. of Library & Info. Sci.
`waseda`	Waseda Univ.
`yamagata-u`	Yamagata Univ
`yamanashi`	Yamanashi Univ.

2. Part of the `fj` news groups are distributed outside the country as requested.

Table 2: Second level domains other than universities in JUNET

adin	asahi	ascii
asp	asr	astd
astec	atr	att-j
cac	canon	canopus
casio	cec	citoh
crl	csk	dcl
dec-j	decjrd	denken
dit	dnp	edr
etl	firmware	foretune
fujitsu	fujixerox	gctech
hitachi	hst	ibmtrl
icm	icot	ipa
jip	jsd	jus
jusoft	k3	kaba
kajima	kcs	kddlabs
kiic	kk	kubota
kyocera	m-giken	m-tsrd
matsubo	meiosk	melco
minpaku	mita	mri
msr	nacsis	ncc
ndg	nec	nig
nts	ntt	oki
omron	pentel	recruit
ricoh	riken	roland
rtri	sanyo	seclab
sharp	sigma	sony
sonytek	soum	sra
sumikin	sun-j	toshiba
tytlabs	uclosk	unisys
yamaha	yhp	ysc

alphabetical representation of Japanese language, however, even among these articles most of them are re-posted messages from various mailing-lists. The major JUNET sites handle about 22 M bytes of articles in a month; 4 M bytes of `fj` news groups and 18 M bytes of USENET news groups.

One of the systems in the JUNET backbone handles about 200 M bytes of information in a month, and 85 percent of them are for the network news and 15 percent are for electronic mails. Since the network news are compressed to half their size before the actual transmission, about 370 M bytes of text information is handled in one of the most busy systems in JUNET.

The international information exchanges are served by two gateways of JUNET; one is in University of Tokyo and named `ccut.cc.u-tokyo.junet`. Another is `kddlab.kddlabs.junet` of KDD laboratories. The `ccut.cc.u-tokyo.junet` is on CSNET as `utokyo-relay` and is operated as the JUNET-CSNET gateway, on the other hand, the `kddlab.kddlabs.junet` is widely known as one of the backbone sites in the UUCPNET and is operated as the JUNET-UUCPNET/USENET gateway.

The amount of international messages can be estimated by the total traffic at these two gateways. The example traffic of June 1987 is shown in Table 3.

Table 3: International Message Traffics

gateway	mail	news
kddlabs.junet	10 MB	18MB
u-tokyo.junet	13MB	0
total	23MB	18MB

In summary, JUNET currently has approximately 41 M bytes of international traffic in a month, and it has been increasing about 3 M bytes per month in the last six months.

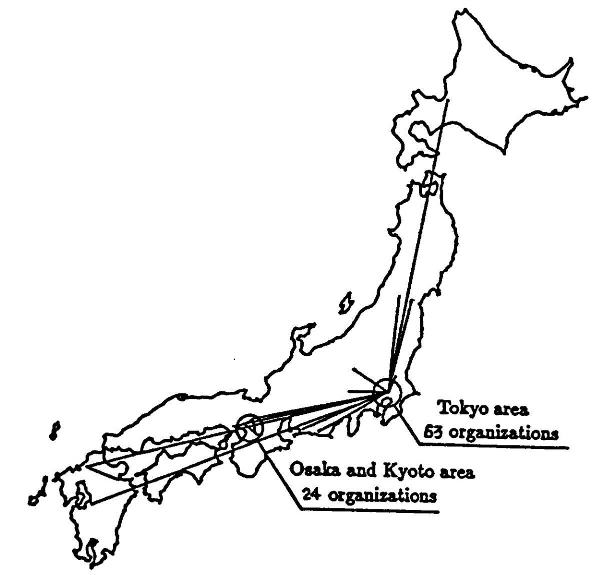


Figure 1: JUNET Geographical Map

JUNET Domain Addressing

In the hierarchy of JUNET domain structure, a domain called 'junet' is the top domain, although we are now preparing to employ ISO's country code (ISO 3116)[9] for Japan 'jp' as the top domain name[10]. The second level domains are called sub-domains, and each of them represents a name of an institute or an organisation. Lower level domains than the sub-domains may be determined at each of the sub-domains. In any cases, the lowest level domains are the names of hosts. The names of sub-domains usually are names well known to the society, but such names sometimes differ in intra/inter-national environment.

Therefore, one or more names can be registered as synonyms for a sub-domain name to help users to address with general knowledge on the name of organisations. A name of a resource, thus, is defined in one of the domains.

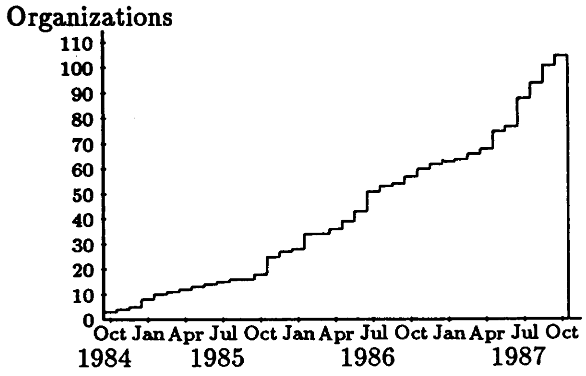


Figure 2: Number of domains

There is one of the distributed name server in each of the domains which handles definitions and deletions of names using a database dedicated to that domain. A name server of a domain thus has a database to define names of lower level domains adjacent to the domain, or names of resources, such as names of mailboxes. The information held by each of the distributed name server is used in retrieving information of resource names and in accessing resources. By this concept, a resource can be defined in a logical domain; a mailbox can be defined even in the top domain, `'junet'`. This provides a name space which is well-matched to the naming concept of the real world yet providing consistency and efficiency of operations.

Design of JUNET message delivery system

There exists a name server for each domain where a distributed resource name can be defined. There is at least one name server in every host; a name server for a domain representing that host. Other than that one, name servers which represent domains located along a path, from the root to this system in the domain tree, can exist in this system. Thus, name servers for the logical domains are managed by entities which are executed in a distributed manner.

A message delivery system using the above concept is implemented using sendmail[5] whose rules are generated by a rule generating system which plays a role of name servers of the JUNET naming concept.

Since the production rules of the sendmail system are different from site to site, the rules have to be generated at each site. To keep the consistency in the rules over JUNET sites, a configuration system to generate the necessary rules was designed and implemented. The configuration system to construct a name server receives information about domain names and about connections for communication among the name server. Then it generates sendmail rule as its output.

The domain database contains relations between the physical connections to the neighbour sites and the domain names which should be solved at that system. Other than the sendmail, the *rmail* command which receives messages through UUCP links was modified to handle JUNET addresses efficiently.

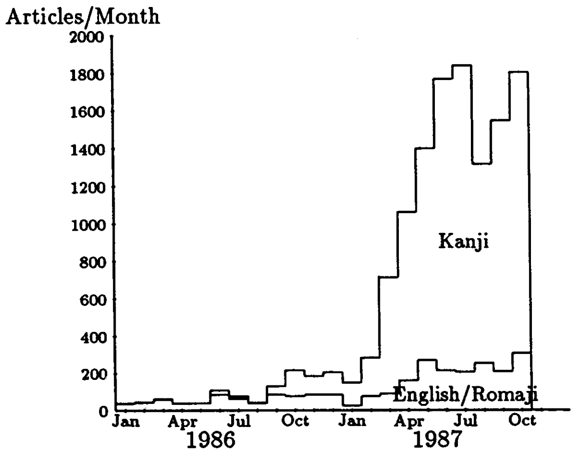


Figure 3: Number of Articles in `fj` newsgroups

International Information Exchange

As described in this paper, the number of messages exchanged in JUNET has increased rapidly, and the number of messages exchanged internationally via the two gateways has also increased as well. Users of other networks, however, sometimes complain about the insufficient information on world networks generated from JUNET. The primary reason for the complaints is obviously caused by the preference of Japanese characters with JUNET users.

The development of general purpose software to handle Japanese characters in non-special hardware environment as well as development of the hand-made *Kanji* fonts encourages us to distribute JUNET *Kanji* messages to other countries. The experimental delivery of the domestic news groups abroad was started to some universities in 1986.

On the other hand, submission of news articles from the USENET environment to JUNET can be achieved by adding a news group called `fj.misc` to the news group list of an article. This is efficient because all of the JUNET sites are subscribing `fj` news groups while some sites are not subscribing USENET news groups. The news article posted this way is handled as a JUNET news article inside Japan.

It is known that the addresses whose top domains are 'junet' are properly handled at both `relay.cs.net` and `uunet.uu.net`. Therefore,

- *user%domain.*`junet@relay.cs.net`
- *user%domain.*`junet@uunet.uu.net`

are the most popular style of addresses from the ARPA Internet name space.

However, both international links are restricted links; mail to/from a JUNET user who does not register the addresses to the gateway cannot be served. This is due to the cost of the international message exchanges. Furthermore, the mails to/from the non-university users in JUNET cannot be passed through the link between `ccut.cc.u-tokyo.junet` and `relay.cs.net`. Therefore, you should check that your friends in JUNET are registered in the gateways before sending them mail. Any questions should be mailed to the JUNET administrators: `junet-admin@junet`.

Domestic Language Support

One of the primary philosophies of JUNET developments is the pursuit of the better computer communications. For that purpose, the native language supports on the network environment has been chosen as one of the goals. The statistics in Figure 3 show that most of the news articles are written in Japanese characters, or in *Kanji* codes. It is observed that a large amount of electronic mail is also exchanged using *Kanji* codes. Therefore, one of the remarkable characteristics of JUNET among academic networks in the world is *Kanji* message handling for text message exchanges. In order to discuss this topic, some basic concepts about Japanese character handling have to be discussed.

Kanji code

The history of *Kanji* code handling in computers is rather short, and there is still some confusion about the *Kanji* code itself. That is to say, several different '*Kanji* code standards' are actually used in the computer world. However, all the 'standards' refer to a single standard defined by JIS (Japanese Industrial Standard) X0208 as their way of defining a set of *Kanji* characters. JIS is always referred to because it can be introduced from any codes defined by International Standard Organisation using ISO 2022 extension guidelines. It defines as *Kanji* code by two 7-bit bytes without using the most significant bits (MSBs). Several different computer *Kanji* codes exist because the switching method defined by ISO 2022 is not practical for random access to text messages.

The JIS X0208 defines not only *Kanji* characters, but also English alphabets, digits, two types of 50 *Kana* characters (phonetic representation of the Japanese language), and special characters. Among them, *Kanji* characters defined by JIS X0208 are divided into two separate groups; one is called level one and another is called level two. The level one includes about 3500 *Kanji* characters and this set provides a sufficient number of *Kanji* characters for most ordinary texts such as technical writings. Although more complicated *Kanji* characters are needed for advanced text applications such text that includes names of people and places, or for literary text. For the purposes such as these, JIS defines a level two *Kanji* character set which includes another 3400 characters.

In total, about 7000 *Kanji* characters are necessary for providing Japanese character capability on a computer. Obviously, this needs more than one byte to represent it, and representing a single character with two bytes is a standard concept. The important issue here is that we still need to use ASCII codes in computers as well as *Kanji* codes. This means we have to establish a way to handle a mixture of ASCII codes and *Kanji* codes. In order to solve this problem, there has to be a way to distinguish ASCII code sequences from *Kanji* code sequences.

There are three major methods which can be used to distinguish *Kanji* sequences from ASCII sequences representating English alphabets and special characters:

1. JIS X0208 codes with JIS X0202 (ISO2022)
2. Extended UNIX Code (EUC)
3. Microsoft *Kanji* Code (Shift-JIS)

Method 1 works by a surrounding a *Kanji* sequence by a designating escape sequence (`ESC-$-@` or `ESC-$-B`) and a sequence of English alphabetics be a designating escape sequence (`ESC-(-J` or `ESC-(-B`).

2. was originally defined by AT&T UNIX Pacific to provide an internationalised version of UNIX

operating system[11] and is becoming to be a standard way to representing *Kanji* codes in the UNIX operating system in Japan. In the EUC, the MSB's of both bytes in a single *Kanji* character are set to 1 whereas the MSB is cleared in an ASCII character..

3. was originally defined for CP/M on personal computers by a Japanese subsidiary of Microsoft Corporation. This is now a *de-facto* standard for personal computers and is also supported in some of the Japanised UNIX environments. In the Microsoft *Kanji* codes, *Kanji* characters are mapped by some function so that first byte is in ranges of `0x81` — `0x9f` and `0xe0` — `0xff`.

In order to cooperate with the ISO standard method of handling character codes, and to utilise existing software which sometimes strips the MSBs off, we decided to use JIS X0208 two 7-bit codes surrounded with escape sequences as the network standard. This decision requires conversion functions from JIS X0208 to local operating system *Kanji* codes, namely EUC/JAPAN and Microsoft *Kanji* code.

Kanji code handling

In order to design network applications using *Kanji* codes, the following discussion has to be made regarding the average environments of existing JUNET systems.

1. There are several kinds of character codes including above codes which are actually used in operating systems as internal codes to represent Japanese characters including *Kanji* characters. Among them, JIS X0208 is not practical for internal code because of the complicated operations of switching modes when seeking a character in a byte sequence; random access to a sequence of byte is impossible. Thus, two 8-bit codes without escape sequence is preferable to two 7-bit codes with escape sequences as internal code for an operating system.
2. Since we have decided to us JIS X0208 as the JUNET network standard *Kanji* code, we have to provide conversion mechanisms from JIS X0208 to internal code of operating systems such as EUC and to Microsoft *Kanji* code.

JUNET approaches

By assuming the above issues, we have developed the following environment for JUNET:

Kanji code As we have described before, we are using JIS X0208 with ISO 2022 / JIS X0202 extension guide lines as a network standard *Kanji* code. Escape sequence introducing JIS X 0208 can either be `ESC-$-@` or `ESC-$-B`. These are two definitions of two minor versions of JIS X0208 and both are legal in a sense of standard definition. Escape sequence introducing ASCII code is `ESC-(-B` and introducing ROMAN character code of JIS is `ESC-(-J`. Only a few characters are different between the ASCII and ROMAN character sets: \ and ˜ in ASCII are replaced by ¥ and ¯ in ROMAN respectively. We therefore treat both of them equally. Note that the default code set for JUNET text message is ASCII code; there is no introducing escape sequence necessary if a text starts with a ASCII code.

Control characters According to the ISO 2022 / JIS X0202, any characters appear in both *Kanji* sequence and ASCII sequence, however, deep backtracking maybe necessary to determine the character mode when a file is accessed randomly. So we decided not to allow control characters appear within *Kanji* code sequence; they can only appear in ASCII code sequence. This rule contributes to make software which handles text messages to be simple and transparent in terms of internationalising because a function to find a control character can transparently be defined.

Single byte Kana code There are another code set for representing Japanese character in a single byte; JIS X0201. In this code, only *Kana* characters are represented. This code set used to be used in mainframes because Japanising of software was easy. Since we now have *Kanji* code set anyway and all the *Kana* representations are included ijn JIS X 0208 using 2 bytes, we eliminate usage of JIS X 0201 to avoid the complexity caused from handling of three different code sets at a time.

Network Software In order to provide an environment for Japanese text message capabilities in JUNET, we have modified the following software to adopt the above strategies:

- Bnews[12] was modified to pass the escape sequences which used to be stripped off in the original version. In the standard implementation, the articles spooled are still represented in network code and the conversion functions to major internal *Kanji* codes are added. Other network news interfaces such as rn and vn have been modified, too.

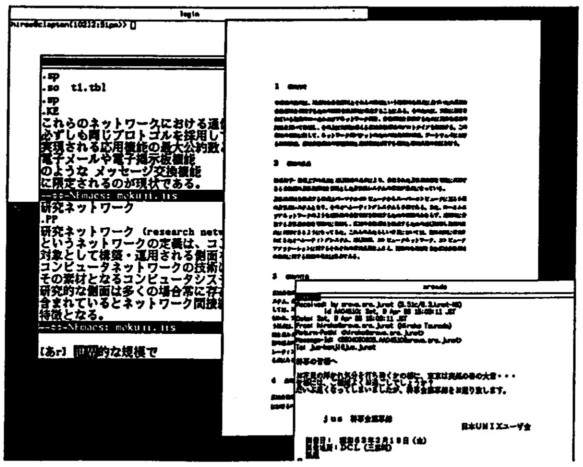
- MH[13] has been added the code conversion facility between the network standard code and the local code.

- GNUemacs and MicroEmacs were modified to handle *Kanji* codes so that we can edit Japanese mail and news articles.

- X Window System was also modified to represent *Kanji* characters on X.V10R3 and X.V10R4. Unfortunately, there were several different approaches to the modifications and offered us no interoperability. In order to establish an interoperability for X.V11. a group of researchers is organised with a mailing list in JUNET[14] for the purpose, and the recent discussions on the list have been very active. The actual work on X.V11 has been completed and is now in the distribution tape of X Window System as a contributed software.

  Level one *Kanji* fonts have been *hand-made* and posted to JUNET so that a user can read and write *Kanji* characters without special *Kanji* terminals.

An example to show the example JUNET environment of Japanese character handling is shown in Figure 4.



```
.sp
.so  t1.tbl
.sp
.KE
これらのネットワークにおける通
必ずしも同じプロトコルを採用し
実現される応用機能の最大公約数
電子メールや電子掲示板機能
のような メッセージ交換機能
に限定されるのが現状である。
--**-NEmacs: mokuji.trs
研究ネットワーク
.PP
研究ネットワーク (research netw
というネットワークの定義は、コ
対象として構築・運用される側面
コンピュータネットワークの技術
その素材となるコンピュータシス
研究的な側面は多くの場合常に存
含まれているとネットワーク間接
特徴となる。
--**-NEmacs: mokuji.trs

[あr] 世界的な規模で
```

Figure 4: Example of JUNET Kanji Environment

Dial-up Links

Fast UUCP Links

The Primary goal of the development of JUNET as the first step was to construct a network providing functions for text message exchange over research communities in Japan. Since JUNET is a volunteer project to provide basis of researches on network communication, UUCP protocol over dial-up links was our choice to start the network because of the popular, easy to set-up and inexpensive features of UUCP technology. The dial-up modems used have been upgraded from V.21 300bps to V.22bis 2400bps and we have almost succeeded in eliminating the V.21 and V.22 1200bps modems.

However, UUCP over 2400bps communication links is not practical as the size of the network has grown and traffic has increased. One of the problems is its slow speed, and it can be reduced by using the high speed modems such as Telebit's Trailblazer or Microcom's AX/9624c. In order to make use of those high speed modems, the flow control mechanism between the modems and the hosts is required and f-protocol UUCP is well known for Xon/Xoff flow control which is one of the popular flow control mechanisms. It is good for X.25 links but is not efficient for the high speed modems because f-protocol UUCP does 7-bit encoding. Instead of f-protocol UUCP, j-protocol UUCP which assumes 8-bit transparent link with Xon/Xoff flow control has been developed. To reduce the overhead in the tty device driver, a new line discipline called `UTTYDISC` which handles only flow control has also been developed. With the combination of j-protocol UUCP and `UTTYDISC`, we can get more than 13Kbps of transmission rate using TrailBlazers or AX/9624c's. The advantage of new UUCP is not only its high performance but also reduction of the tty port occupation.

Dial-up Links — Motivation

The UUCP link is intended for file transfer in batch mode and the file systems at the destination machines are liable to overflow. This problem is serious; when a file system overflows, many new articles and even mails are lost. Therefore, reliable datalink protocols which replace UUCP efficiently are preferable.

As JUNET is a volunteer based network and as we do not have enough funds to get dedicated lines between organisations, we have to make use of the dial-up lines at least for several months. X.25 packet switching network is one of the candidates. We can pass the IP datagrams over X.25 links as defined in [15], however, it is not practical for the small organisations and for the newly coming organisations because of its cost. Another candidate is to pass the IP datagrams over the dial-up links.

Providing this facility, we can make the network more reliable without any additional hardware. We have developed a serial KIP module for dial-up links, called DL, derived from Rick Adams' SLIP.

Dial-up IP Link — Structure

IP connections over a dial-up link have been usually considered for personal computers at home[16], however, they are still effective for easy construction of an internetworking among universities, especially on the achieved performance with the high-speed modems. The system for dial-up IP link thus was designed and developed on Sun OS version 3.2 which is compatible with 4.2BSD UNIX.

The system consists of three modules. The *dldriver* resides in the kernel and it receives and sends IP datagrams with the IP module which also exists in the kernel space. It has communication entities as a 'special file' in the UNIX file space called `/dev/dlx`.

The major portion of the system, *dldaemon*, is a permanent user process which reads information from the *dldriver* and controls it through `/dev/dlz`. The *dldaemon* is invoked at a boot time and resides on the system permanently. As soon as the process is initiated, it registers a network interface (*if*) called `dlz` and sets up the routing information in the kernel space so that the IP module can pass IP fragments with addresses routed toward the link to the *dldriver*.

The *dldriver* includes a tty line discipline called `DLIPDISC`, which does character mapping to escape `DC1/DC3` and other control characters. Another module called *dlattach*, which is a user command, is invoked at login time to pass the information about the serial port to the *dldaemon*. Figure 5 illustrates the structure of the system.

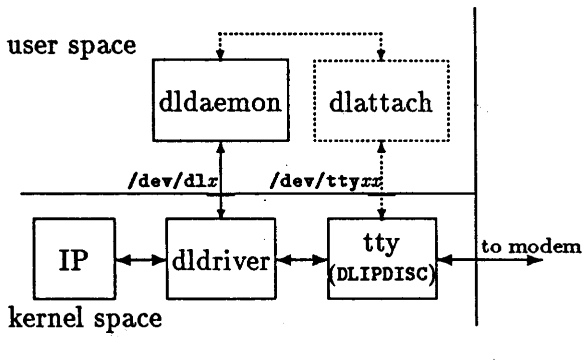


Figure 5: Structure of the DL

A connection established with the `DL` is a point-to-point connection between a master system which initiates the connection and a slave system. The following example of sequence shows an overview of the procedure and the functions of the system:

1. When an `IP` fragment is selected to the `dlz` interface by the routing function in the `IP` module, the *dldriver* passes the fragment to the attached tty line if the link has already been established. Otherwise, the *dldriver* requests the *dldaemon* to initiate dialing to the remote site.
2. The master *dldaemon* logs-in to the slave site and invokes the *dlattach* command.
3. The *dlattach* provides the *dldaemon* residing on the slave site with the information about the serial port and about the caller's name, as shown in the dotted line of Figure 5.
4. Then, each of the *dldaemons* at the both systems issues a system call to switch a line discipline to `DLIPDISC`. Finally, they complete establishment of a dial-up link ready for IP communication between the systems.
5. A connection is terminated by the *dldaemon* when it detects that 60 seconds has passed without any IP fragment transmissions.

The measured performance of the system with two TrailBlazers is approximately 7.5Kbps and two AX/9624c is approximately 8Kbps for end-to-end data transmission by ftp. Furthermore, we can obtain the performance of more than 10Kbps with an UDP-based file transfer protocol whose window size is large enough to cover the delay originated from the internal protocol of the modems.

Discussions

The DL have the following characteristics compared to other media:

1. Link is usually dead and activated by requests.
2. Dialing takes 20 — 60 seconds. The first datagram takes this long delay before reaching the destination.
3. Link speed is not so high. The propagation delay is relatively large because of the modems' internal protocol.
4. Link cost depends on the connection time rather than the amount of the traffic.

DL as a replacement of UUCP

The primary usage of DL is a replacement of UUCP. The messages are transferred in batched manner, however, each IP fragment is carried over the DL as in the dedicated lines. More precisely, *uucp* and *uux* replacements receive the file transfer requests and put the data once onto the spool directory. When the connection is successfully established, the processes which talk to the remote daemon with SMTP or NNTP are invoked. We have modified version of *nntpd* and a client process named *sendnntp*, and we can prevent the file system from overflowing.

The timing when the dial operation is initiated may be determined by an evaluation function depending on the amount of the data spooled for a particular destination, the elapsed time since last connection, and so on. The *dldaemon* executes such a function periodically, and determines whether it should initiate dialing.

Advanced usage of DL

For the DL as a replacement of UUCP, the existence of the link do not have be to inform to other sites. But the processes on the host other than the gateways want to send/receive the IP datagrams while the link is established. For this requirement, the gateways should propagate the routing information when the DL is established or closed.

Unlike the permanent network links, the DL is usually closed and initiated by the arrival of the IP datagram to the remote network. This means that the DL should be handled as 'active' in the routing information even the actual connection is not established. A process on a site other than gateway can issue a TCP connection request according to 'active' state in the routing information, however, it may time out because the dial operation may take 20-60 seconds. In such a case, the DL connections is established, while the TCP has been timed out and no actual data transmission is carried out.

We use the high speed modems for the DL to obtain good performance, however, those modems have relatively large delay time and we cannot expect the maximum performance with the ordinal TCP window size. Using a file transfer protocol with large window size over UDP, the performance of the DL exceeds 10Kbps, whereas 8Kbps with ftp.

In order to solve these problems and achieve the best use of the DL, a new routing procedure and congestion control technique suitable for the DL is now developing. The routing information should contain whether the path to a destination includes the DL or not. The time out mechanism and the window size determination mechanism in the TCP module are affected to such a information in the routing information. Apart from the efficiency problems, the researches on the security issue should also be worked to use the DL practically.

Conclusion

Development of JUNET started in October 1984. Since then various researches on computer networking and distributed environment have been actively done as the rapid growth in the size of the network. Among them, name management functions to construct a hierarchical domain name space, Japanese character handling, and communication technologies using the high-speed modems have been focused as primary research concerns.

The actual work for the addressing and routing of JUNET text messages is achieved by a name server concept and its implementation. This system receives control messages to specify the information about logical domain name, connections and methods to deliver messages, and generates a sendmail rule set. This software provides an environment where the logical naming definitions and physical routing issues are clearly separated so that reliability, efficiency, extensibility and flexibility of communication in the network are simultaneously achieved.

Internationalisation of computer software is one of the most important issues in JUNET and its communication software. The clear separation of network *Kanji* code and internal operating system code employed in JUNET software provides a transparent environment on Japanese character handling in computer networks. Availability of Japenese messages obviously encourages JUNET users very much to exchange messages over the network.

A new UUCP protocol and tty driver enhancement are developed for the requirements of higher transmission rate over the dial-up lines. As the result, more than 13Kbps UUCP transmission rate is achieved. This encouraged us to migrate onto TCP/IP suite using dial-up lines as well as using leased lines. Performance of the implementation of the dial-up IP link is about 8Kbps which is practical enough to construct a distributed environment over a widely interconnected network environment.

Progresses of JUNET technologies discussed in this paper lead us to many topics of future studies such as:

- Enhancement of name servers which handles general distributed resources.
- Establishment of gateway technologies such as optimal routing strategies based upon constructions of IP-based network using the dial-up IP link and IP over leased lines.
- Supports of multi-media message exchanges enhances the environment with multi-language supports currently achieved.

The technologies developed in JUNET can generally be used to construct a network interconnection with inexpensive cost.

Acknowledgements

The authors would like to thank Youichi Shinoda, Keisuke Tanaka and Hiroshi Tachibana for their efforts in developing software. The dial-up IP link was achieved as a result of discussions with members of the JUNET project, especially with Susumu Sano.

References

[1] Jun Murai and Akira Kato. Researches in Network Development of JUNET. In *Proceedings of SIGCOMM '87 Workshop*, ACM, 1987.

[2] J. Murai and T. Asami. A network for research and development communications in Japan — JUNET —. In *Proceedings of First Pacific Computer Communications Symposium*, 1985.

[3] J.S. Quaterman and J.C. Hoskins. Notable Computer Networks. *CACM*, 29(10), October 1986.

[4] D.A. Nowitz and M.E. Lesk. *A Dial-Up Network of UNIX Systems*. Technical Report, Bell Telephone Laboratories, August 1987.

[5] Elic Allman. Sendmail — An Interconnecting Mail Rerouter, Version 4.2. In *UNIX Programmer's Manual, 4.2 Berkeley Software Distribution*, Univ. of California, Berkeley, 1983.

[6] J. Postel (ed.). Internet Protocol — DARPA Internet Program Protocol Specification. RFC 791. 1980.

[7] J. Postel (ed.). Transmission Control Protocol — DARPA cInternet Program Protocol Specification. RFC 793. 1981.

[8] D. Comer. The computer science research network CSNET: A history and status report. *CACM*, 26(10), October 1983.

[9] ISO. Codes for the Representation of Names of Countries. ISO 3116. 1981.

[10] J. Postel. Domain requirements. RFC 920. 1984.

[11] Hiromichi Kogure and Richard McGowan. A UNIX System V STREAMS TTY Implementation for Multiple Language Processing. In *USENIX Summer Conference Processings*, USENIX, 1987.

[12] S.L. Emerson. USENET: A bulletin board for UNIX users. *BYTE*, September 1983.

[13] R.S. Gaines, S. Borden and N.Z. Shapiro. *The MH Message Handling System: User's Manual.* Rand Corporation, 1979.

[14] Hiroshi Tachibana. PD *Kanji* font (tools and ascii fonts). Network news posted to `fj.sources` as `<1676@rika.cs.titech.JUNET>`, July 1987.

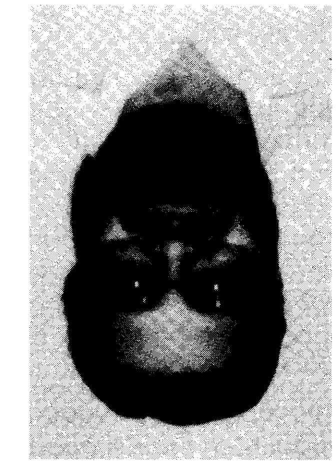
[15] J.T. Korb. A Standard for the transmission of IP Datagrams Over Public Data Networks. RFC 877. 1983.

[16] D.J. Farber, G.S. Delp and T.M. Conte. A Thinwire Protocol for connecting personal computers to the INETERNET. RFC 914. 1984.

Cake: a Fifth Generation Version of make

Zoltan Somogyi
UUCP: {uunet,mcvax,ukc}!munnari.oz!zs
zs@mulga.oz.au, zs@mulga.uucp
ARPA: zs%munnari.oz@uunet.uu.net
CSNET: zs%munnari.oz@australia

Department of Computer Science
University of Melbourne
Parkville, 3052 Victoria, Australia



Zoltan is a grad stude hacker droid who haunts the night corridors of the University of Melbourne. He is interested in Life, the Universe, and Everything. His main research interest is in parallel logic programming, but he sometimes cobbles together useful software in his spare time.

Zoltan lives in Australia.

Abstract

Make is a standard UNIX utility for maintaining computer programs. Cake is a rewrite of make from the ground up. The main difference is one of attitude: cake is considerably more general and flexible, and can be extended and customised to a much greater extent. It is applicable to a wide range of domains, not just program development.

Introduction

The UNIX utility make (Feldman, '79) was written to automate the compilation and recompilation of C programs. People have found make so successful in this domain that they do not wish to be without its services even when they are working in other domains. Since make was not designed with these domains in mind (some of which, e.g., VLSI design, did not even exist when make was written), this causes problems and complaints. Nevertheless, implied in these complaints is an enormous compliment to the designers of make; one does not hear many grumbles about programs with only a few users.

The version of make described in (Feldman, '79) is the standard utility. AT&T modified it in several respects for distribution with System V under the name augmented make (AT&T, '84). We know of two complete rewrites: enhanced make (Hirgelt, '83) and fourth generation make (Fowler, '85). All these versions remain oriented towards program maintenance[1].

Here at Melbourne we wanted something we could use for text processing. We had access only to standard make and spent a lot of time wrestling with makefiles that kept on getting bigger and bigger. For a while we thought about modifying the make source, but then decided to write something completely new. The basic problem was the

1. Since this paper was written, two other rewrites have come along: mk (Hume, '87) and nmake.

inflexibility of make's search algorithm, and this algorithm is too embedded in the make source to be changed easily.

The name cake is an historical accident. Cake follows two other programs whose names were also puns on make. One was bake, a variant of make with built-in rules for VLSI designs instead of C programs (Gedye, '84). The other was David Morley's shell script fake. Written at a time when disc space on our machine was extremely scarce, and full file systems frequently caused write failures, it copied the contents of a directory to /tmp and invoked make there.

The structure of the paper is as follows. Section 2 shows how cake solves the main problems with make, while section 3 describes the most important new features of cake. The topics of section 4 are portability and efficiency. The paper assumes that you have some knowledge of make.

The problems with make

Make has three principal problems. These are:

1. It supports only suffix-based rules.
2. Its search algorithm is not flexible enough.
3. It has no provisions for the sharing of new make rules.

These problems are built deep into make. To solve them we had to start again from scratch. We had to abandon backward compatibility because the make syntax is not rich enough to represent the complex relationships among the components of large systems. Nevertheless, the cake user interface is deliberately based on make's; this helps users to transfer their skills from make to cake. The *functionalities* of the two systems are sufficiently different that the risk of confusion is minimal[2].

Probably the biggest single difference between make and cake lies in their general attitudes. Make is focused on one domain: the maintenance of compiled programs. It has a lot of code specific to this domain (especially the later versions). And it crams all its functionality into some tight syntax that treats all sorts of special things (e.g., .SUFFIXES) as if they were files.

Cake, on the other hand, uses different syntax for different things, and keeps the number of its mechanisms to the minimum consistent with generality and flexibility. This attitude throws a lot of the functionality of make over the fence into the provinces of other programs. For example, where make has its own macro processor, cake uses the C preprocessor; and where make has special code to handle archives, cake has a general mechanism that *just happens* to be able to do substantially the same job.

Only suffix-based rules

All entries in a makefile have the same syntax. They do not, however, have the same semantics. The main division is between entries which describe simple dependencies (how to make file a from file b), and those which describe rules (how to make files with suffix .x from files with suffix .y)[3]. Make distinguishes the two cases by treating as a rule any dependency whose target is a concatenation of two suffixes.

For this scheme to work, make must assume three things. The first is that all interesting files have suffixes; the second is that suffixes always begin with a period; the third is that prefixes are not important. All three assumptions are violated in fairly common situations. Standard make cannot express the relationship between file and file.c (executable and source) because of assumption 1, between file and file,v (working file and RCS file) because of assumption 2, and between file.o and ../src/file.c (object and source) because of assumption 3. Enhanced make and fourth generation make have special forms for some of these cases, but these cannot be considered solutions because special forms will always lag behind demand for them (they are embedded in the make source, and are therefore harder to change than even the built-in rules).

Cake's solution is to do away with make-style rules altogether and instead to allow ordinary dependencies to function as rules by permitting them to contain variables. For example, a possible rule for compiling C programs is

2. This problem, called cognitive dissonance, is discussed in Weinberg's delightful book (Weinberg, '71).

3. For the moment we ignore entries whose targets are special entities like .IGNORE, .PRECIOUS, etc.

```
%.o:     %.c
         cc -c %.c
```

where the % is the variable symbol. This rule is actually a *template* for an infinite number of dependencies, each of which is obtained by consistently substituting a string for the variable %.

The way this works is as follows. First, as cake seeks to update a file, it matches the name of that file against all the targets in the description file. This matching process gives values to the variables in the target. These values are then substituted in the rest of the rule[4]. (The matching operation is a form of *unification*, the process at the heart of logic programming; this is the reason for the *fifth generation* bit in the title.)

Cake actually supports 11 variables: % and %0 to %9. A majority of rules in practice have only one variable (canonically called %), and most of the other rules have two (canonically called %1 and %2). These variables are local to their rules. Named variables are therefore not needed, though it would be easy to modify the cake source to allow them.

Example
If cake wanted to update prog.o, it would match prog.o against %.o, substitute prog for % throughout the entry, and then proceed as if the cakefile contained the entry

```
prog.o: prog.c
         cc -c prog.c
```

This arrangement has a number of advantages. One can write

```
%.o:     RCS/%.c,v
         co -u %.c
         cc -c %.c
```

without worrying about the fact that one of the files in the rule was in a different directory and that its suffix started with a nonstandard character. Another advantage is that rules are not restricted to having one source and one target file. This is useful in VLSI, where one frequently needs rules like

4. After this the rule should have no unexpanded variables in it. If it does, cake reports an error, as it has no way of finding out what the values of those variables should be.

```
%.out:  %.in %.circuit
   simulator %.circuit < %.in > %.out
```

and it can also be useful to describe the full consequences of running yacc

```
%.c %.h:  %.y
          yacc -d %.y
          mv y.tab.c %.c
          mv y.tab.h %.h
```

Inflexible search algorithm

In trying to write a makefile for a domain other than program development, the biggest problem one faces is usually make's search algorithm. The basis of this algorithm is a special list of suffixes. When looking for ways to update a target file.x, make searches along this list from left to right. It uses the first suffix .y for which it has a rule .y.x and for which file.y exists.

The problem with this algorithm manifests itself when a problem divides naturally into a number of stages. Suppose that you have two rules .c.b and .b.a, that file.c exists and you want to issue the command make| file.a. Make will tell you that it doesn't know how to make file.a. The problem is that for the suffix .b make has a rule but no file, while for .c it has a file but no rule. Make needs a *transitive rule* .c.a to go direct from file.c to file.a.

The number of transitive rules increases as the square of the number of processing stages. It therefore becomes significant for program development only when one adds processing stages on either side of compilers. Under UNIX, these stages are typically the link editor ld and program generators like yacc and lex. Half of standard make's built-in rules are transitive ones, there to take care of these three programs. Even so, the builtin rules do not form a closure: some rare combinations of suffixes are missing (e.g., there is no rule for going from yacc source to assembler).

For builtin rules a slop factor of two may be acceptable. For rules supplied by the user it is not. A general-purpose makefile for text processing under UNIX needs at least six processing stages to handle nroff/troff and their preprocessors lbl, bib, pic, tbl, and eqn, to mention only the ones in common use at Melbourne University.

Cake's solution is simple: if file1 can be made from file2 but file2 does not exist, cake will try to *create* file2. Perhaps file2 can be made from file3, which can be made from file4, and so on, until we come to a file which does exist. Cake will give up only when there is *absolutely no way* for it to generate a feasible update path.

Both the standard and later versions of make consider missing files to be out of date. So if file1 depends on file2 which depends on file3, and file2 is missing, then make will remake first file2 and then file1, even if file1 is more recent than file3.

When using yacc, we frequently remove generated sources to prevent duplicate matches when we run egrep ... *.[chyl]. If cake adopted make's approach to missing files, it would do a lot of unnecessary work, running yacc and cc to generate the same parser object again and again[5].

Cake solves this problem by associating dates even with missing files. The *theoretical update time* of an existing file is its modify time 7 given by stat(2)); the theoretical update time of a missing file is the theoretical update time of its youngest ancestor. Suppose the yacc source parser.y is older than the parser object parser.o, and parser.c is missing. Cake will figure that if it recreated parser.c it would get a parser.c which *theoretically* was last modified at the same time as parser.y was, and since parser.o is younger than parser.y, theoretically it is younger than parser.c as well, and therefore up-to-date.

No provisions for sharing rules

Imagine that you have just written a program that would normally be invoked from a make rule, such as a compiler for a new language. You want to make both the program and the make rule widely available. With standard make, you have two choices. You can hand out copies of the rules and get users to include it in their individual makefiles; or you can modify the make source, specifically, the file containing the built-in rules. The first way is error-prone and quite inconvenient (all those rules cluttering up your makefile when you should never need to even look at them). The second way can be impractical; in the development stage because the rules can change frequently and after that because you want to distribute your program to sites that may lack the make source. And of course two such modifications may conflict with one another.

5. In this case make is rescued from this unnecessary work by its built-in transitive rules, but as shown above this should not be considered a *general* solution.

Logically, your rules belong in a place that is less permanent than the make source but not as transitory as individual makefiles. A library file is such a place. The obvious way to access the contents of library files is with #include, so cake filters every cakefile through the C preprocessor.

Cake relies on this mechanism to the extent of not having *any* built-in rules at all. The standard cake rules live in files in a library directory (usually /usr/lib/cake). Each of these files contains rules about one tool or group of tools. Most user cakefiles #define some macros and then include some of these files. Given that the source for program prog is distributed among prog.c, aux1.c, aux2.c, and parser.y, all of which depend on def.h, the following would be a suitable cakefile:

```
#define MAIN    prog
#define FILES   prog aux1 aux2 parser
#define HDR     def

#include        <Yacc>
#include        <C>
#include        <Main>
```

The standard cakefiles Yacc and C, as might be expected, contain rules that invoke yacc and cc respectively. They also provide some definitions for the standard cakefile Main. This file contains rules about programs in general, and is adaptable to all compiled languages (e.g., it can handle NU-Prolog programs). One entry in Main links the object files together, another prints out all the sources, a third creates a tags file if the language has a command equivalent to ctags, and so on.

Make needs a specialised macro processor; without one it cannot substitute the proper filenames in rule bodies. Fourth generation make has not solved this problem but it still wants the extra functionality of the C preprocessor, so it grinds its makefiles through both macro processors! Cake solves the problem in another way, and can thus rely on the C preprocessor exclusively.

Standard make's macro facilities are quite rudimentary, as admitted by (Feldman, '79). Unfortunately, the C preprocessor is not without flaws either. The most annoying is that the bodies of macro definitions may begin with blanks, and will if the body is separated from the macro name and any parameters by more than one blank (whether space or tab). `Cake` is distributed with a fix to this problem in the form of a one-line change to the preprocessor source, but this change probably will not work on all versions of UNIX and definitely will not work for binary-only sites.

The new features of `cake`

The above solutions to `make`'s problems are useful, but they do not by themselves enable `cake` to handle new domains. For this `cake` employs two important new mechanisms: dynamic dependencies and conditional rules.

Dynamic dependencies

In some situations it is not convenient to list in advance the names of the files a target depends on. For example, an object file depends not only on the corresponding source file but also on the header files referenced in the source.

Standard `make` requires all these dependencies to be declared explicitly in the `makefile`. Since there can be rather a lot of these, most people either declare that all objects depend on all headers, which is wasteful, or declare a subset of the true dependencies, which is error-prone. A third alternative is to use a program (probably an `awk` script) to derive the dependencies and edit them into the `makefile`. (Walden, '84) describes one program that does both these things; there are others. These systems are usually called `makedepend` or some variation of this name.

The problems with this approach are that it is easy to alter the automatically-derived dependencies by mistake, and that if a new header dependency is added the programmer must remember to run `makedepend` again. The C preprocessor solves the first problem; the second, however, is the more important one. Its solution must involve scanning though the source file, checking if the programmer omitted to declare a header dependency. So why not use this scan to *find* the header dependencies in the first place?

`Cake` attacks this point directly by allowing parts of rules to be specified at run-time. A command enclosed in double square brackets[6] may appear in a rule anywhere a filename or a list of filenames may appear. For the example of the C header files, the rule would be

```
%.o:     %.c [[ccincl %.c]]
         cc -c %.c
```

signifying that `x.o` depends on the files whose names are listed in the output of the command `ccincl x.c` [7], as well as on `x.c`. The matching process would convert this rule to

```
x.o:     x.c [[ccincl x.c]]
         cc -c x.c
```

which in turn would be *command expanded* to

```
x.o:     x.c hdr.h
         cc -c x.c
```

if `hdr.h` were the only header included in `x.c`.

Command patterns provide replacements for `fourth generation make's` directory searches and special macros. `[[find\ <dirs>\ -name\ <filename>\ -print]]` does as good a job as the special-purpose `make` code in looking up source files scattered among a number of directories. `[[basename\ <filename>\ <suffix>]]` can do an even better job: `make` cannot extract the base from the name of an RCS file.

A number of tools intended to be used in just such contexts are distributed together with `cake`. `Ccincl` is one. `Sub` is another: its purpose is to perform substitutions. Its arguments are two patterns and some strings: it matches each string against the first pattern, giving values to its variables; then it applies those values to the second pattern and prints out the result of this substitution. For example, in the example of section 2.3 the `cakefile main` would invoke the command `[[sub\ X\ X.o\ FILES]]`[8], the value of `FILES` being `prog aux1 aux2 parser`, to find that the object files it must link together to create the executable `prog` are

6. Single square brackets (like most special characters) are meaningful to `csh`: they denote character classes. However, we are not aware of any legitimate contexts where two square brackets *must* appear together. The order of members in such classes is irrelevant, so if a bracket must be a member of such a class it can be positioned away from the offending boundary (unless the class is a singleton, in which case there is no need for the class in the first place).

7. `Ccincl` prints out the names of the files that are `#included` in the file named by its argument. Since `ccincl` does not evaluate any of the C preprocessor's control lines, it may report a superset of the files actually included.

prog.o aux1.o aux2.o parser.o.

`Cake` allows commands to be nested inside one another. For example, the command `[[sub\ X.h\ X\ [[ccincl\ file.c]]]]` would strip the suffix `.h` from the names of the header files included in `file.c` [9].

Conditional rules

Sometimes it is natural to say that `file1` depends on `file2` *if* some condition holds. None of the `make` variants provide for this, but it was not too hard to incorporate conditional rules into `cake`.

A `cake` entry may have a condition associated with it. This condition, which is introduced by the reserved word `if`, is a boolean expression built up with the operators `and`, `or` and `not` from primitive conditions.

The most important primitive is a command enclosed in double curly braces. Whenever `cake` considers applying this rule, it will execute this command after matching, substitution and command expansion. The condition will return true if the command's exit status is zero. This runs counter to the intuition of C programmers, but it conforms to the UNIX convention of commands returning zero status when no abnormal conditions arise. For example, `{{grep\ xyzzy\ file}}` returns zero (i.e., true) if xyzzy occurs in `file` and nonzero (false) otherwise.

Conceptually, this one primitive is all one needs. However, it has considerable overhead, so `cake` includes other primitives to handle some special cases. These test whether a filename occurs in a list of filenames, whether a pattern matches another, and whether a file with a given name exists. Three others forms test the internal `cake` status of targets. This status is `ok` if the file was up-to-date when `cake` was invoked, `cando` if it wasn't but `cake` knows how to update it, and `noway` if `cake` does not know how to update it.

As an example, consider the rule for RCS.

8. `sub` uses `X` as the character denoting variables. It cannot use `%`, as all `%`'s in the command will have been substituted for by `cake` by the time `sub` is invoked.

9. As the outputs of commands are substituted for the commands themselves, `cake` takes care not to scan the new text, lest it find new double square brackets and go into an infinite loop.

```
%:      RCS/%,v     if exist RCS/%,v
        co -u %
```

Without the condition the rule would apply to all files, even ones which were not controlled by RCS, and even the RCS files themselves: there would be no way to stop the infinite recursion (`%` depends on `RCS/%,v` which depends on `RCS/RCS/%,v,v`...).

Note that conditions are command expanded just like other parts of entries, so it is possible to write

```
%:  archive   if % in [[ar t archive]]
    ar x archive %
```

The implementation

Portability

`Cake` was developed on a Pyramid 90x under 4.2bsd. At Melbourne University it now runs on a VAX under 4.3bsd, various Sun-3's under SunOS 3.4, an Encore Multimax under Umax 4.2, a Perkin-Elmer 3240 and an ELXSI 6400 under 4.2bsd, and on the same ELXSI under System V. It has not been tested on either System III or version 7.

`Cake` is written in standard C, with (hopefully) all machine dependencies isolated in the makefile and a header file. In a number of places it uses `#ifdef` to choose between pieces of code appropriate to the AT&T and Berkeley variants of UNIX (e.g., to choose between `time()` and `gettimeofday()`). In fact, the biggest hassle we have encountered in porting `cake` was caused by the standard header files. Some files had different locations on different machines (`/usr/include` vs. `/usr/include/sys`), and the some versions included other header files (typically `types.h`) while others did not.

As distributed `cake` is set up to work with `csh`, but it is a simple matter to specify another shell at installation time. (In any case, users may substitute their preferred shell by specifying a few options.) Some of the auxiliary commands are implemented as `csh` scripts, but these are small and it should be trivial to convert them to another shell if necessary.

Efficiency

`Fourth generation make` has a very effective optimisation system. First, it forks and execs only once. It creates one shell, and thereafter, it pipes commands to be executed to this shell and gets back status information via another pipe. Second, it compiles its `makefiles` into internal form, avoiding parsing except when the compiled version is out of date with respect to the master.

The first of these optimisations is an absolute winner. `cake` does not have it for the simple reason that it requires a shell which can transmit status information back to its parent process, and we don't have access to one (this feature is provided by neither of the standard shells, `sh` and `csh`).

`cake` could possibly make use of the second optimisation. It would involve keeping track of the files the C preprocessor includes, so that the `makefile` can be recompiled if one of them changes; this must be done by fourth generation make as well though (Fowler, '85) does not mention it. However, the idea is not as big a win for `cake` as it is for `make`. The reason is as follows.

The basic motivations for using `cake` rather than `make` is that it allows one to express more complex dependencies. This implies a bigger system, with more and slower commands than the ones `make` usually deals with. The times taken by `cake` and the preprocessor are insignificant when compared to the time taken by the programs it most often invokes at Melbourne. These programs, `ditroff` and `nc` (the NU-Prolog compiler that is itself written in NU-Prolog), are notorious CPU hogs.

Here are some statistics to back up this argument. The *overhead ratio* is given by the formula

$$\frac{\textit{cake process system time + children user time + children system time}}{\textit{cake process user time}}$$

This is justifiable given that the `cake` implementor has direct control only over the denominator; the kernel and the user's commands impose a lower limit on the numerator.

We have collected statistics on every `cake` run on two machines at Melbourne, mulga and munmurra[10]. These statistics show that the overhead ration on mulga is 11 while on munmurra it is 86. This suggests that the best way to lower total CPU time is not to tune `cake` itself but to reduce the number of child processes. To this end, `cake` caches the status returned by all condition commands `{{command}}` and the output of all command patterns `[[command]]`. The first cache has hit ratios of 42 and 54 percent on munmurra and mulga respectively, corresponding roughly to the typical practice in which a condition and its negation select one out of a pair of rules. The second cache has a hit ratio of about 80 percent on both machines; these hits are usually the second and later occurrences of macros whose values contain commands.

10. On mulga (a Perkin-Elmer 3240), the main applications are text processing and the maintenance of a big bibliography (over 58000 references). On munmurra (an EXLSI 6400), the main application is NU-Prolog compilation.

`cake` also uses a second optimisation. This one is borrowed from standard `make`: when an action contains no constructs requiring a shell, `cake` itself will parse the action and invoke it through exec. We have no statistics to show what percentage of actions benefit from this, but a quick examination of the standard `cakefiles` leads us to believe that it is over 50 percent.

Overall, `cake` can do a lot more than `make`, but on things which *can* be handled by `make`, `cake` is slightly slower than standard `make` and a lot slower than fourth generation make. Since the main goal of `cake` is generality, not efficiency, this is understandable. If efficiency is important, `make` or one of its other successors is always available as a fallback.

Availability

The `cake` distribution contains the `cake` source, some auxiliary programs and shell scripts (many useful in their own right), diffs for the `lex` driver and the C preprocessor, library cakefiles, manual entries, and an earlier version of this paper (Somogyi, '87). It was posted to the Usenet newsgroup comp.sources.unix in October of 1987.

Acknowledgements

John Shepherd, Paul Maisano, David Morley and Jeff Schultz helped me to locate bugs by being brave enough to use early versions of `cake`. I would like to thank John for his comments on drafts of this paper.

This research was supported by a Commonwealth Postgraduate Research Award, the Australian Computer Research Board, and Pyramid Australia.

References

(AT&T, '84) Augmented version of make, in: *UNIX System V - release 2.0 support tools guide*, AT&T, April 1984.

(Feldman, '79) Stuart I. Feldman, Make - a program for maintaining computer programs, *Software - Practice and Experience*, 9:4 (April 1979), pp. 255-265.

(Fowler, '85) Glenn S. Fowler, A fourth generation make, *Proceedings of the USENIX 1985 Summer Conference*, Portland, Oregon, June 1985, pp. 159-174.

Gedye, '84 David Gedye, Cooking with CAD at UNSW, Joint Microelectronics Research Center, University of New South Wales, Sydney, Australia, 1984.

(Hirgelt, '83) Edward Hirgelt, Enhancing make or re-inventing a rounder wheel, *Proceedings of the USENIX 1983 Summer Conference*, Toronto, Ontario, Canada, July 1983, pp. 45-58.

(Hume, '87) Andrew Hume, Mk: a successor to make, *Proceedings of the USENIX 1987 Summer Conference*, Phoenix, Arizona, June 1987, pp. 445-457.

(Somogyi, '87) Zoltan Somogyi, Cake: a fifth generation version of make, *Australian UNIX system User Group Newsletter*, 7:6 (April 1987), pp. 22-31.

(Walden, '84) Kim Walden, Automatic generation of make dependencies *Software - Practice and Experience*, 14:6 (June 1984), pp. 575-585.

(Weinberg, '71) Gerald M. Weinberg, The psychology of computer programming, Van Nostrand Reinhold, New York, 1971.

Thank you, Zoltan

Zoltan has agreed that the latest version of 'cake' can be put on the next EUUG conference tape. Thank you, Zoltan.

Competitions at the London Conference

As is traditional at EUUG conferences a competition was held. As is not traditional there were two competitions.

The Signal Competition

The other competition was inspired by the excellent replies generated by the *errno* competition a couple of years ago. This time the task was to invent new signals and their meanings.

The winning entry was:

SIGTITANIC Floating point exception

Submitted by Martyn Tovey from BRS Europe.

There were many other entries of high standard. Here is a selection of the best:

Signal	Explanation
SIGHTSEEING	Delegate lost
SIGTUBE	Bus error
SIGINGINTHERAIN	Pipe overflow
SIGFERRY	Data packet has crossed communication channel
SIGTUNNEL	Reserved for future use — will replace SIGFERRY
SIGTUBE	Attempt to pack ≥ bits into a byte
SIGLT	Double sigbus
SIGPEDESTRIAN	Slow data packet is overwritten by expedited data
SIGSIGSIGSIG	Excessive recursion depth
SIGBEKO	Segregation violation
SIGNORINA	36-24-36
SIGNAB	Tax evasion violation
SIGQUIT	You chief programmer just joined another company
SIGSUN	Processor superseded
SIGNAL	Not another language !
SIGJEDI	Use of "The Force" required to continue process
SIGSHUTTLE	Ring failure
SIGNEPHEW	A process, not a child of yours, has died
SIGMUND	Child process too close to motherboard
SIGPROC	Your paper missed the proceedings deadline
SIGNIFICANT	Too many digits
SIGILL	Too much curry
SIGCIA	Are we being bugged
SIGLIONS	I don't understand this
SIGBLAH	Unexpected file recovery
SIGMI5	Filename classified
SIG11	Last order (UK only)
SIGMAINSSPIKE	Incoming signal on power supply
SIGBSD	Your program is using too little memory
SIGCISC	Instruction too complex
SIGPTO	Page fault signal
SIGBLAH	Comment (not) found
SIGGWR	Train arrived on time
SIGMODEM	Unex ect d oss of car ier
SIGWESTEND	American tourist looking for Harrods
SIGLHR	Excessive aircraft noise

SIGNET	Distributed system caught
SIGNUTCRACKER	Broken kernel
SIGAMNESIA	No more memory
SIGSTAB	Et tu IBM
SIGHUME	Large noise source detected
SIGPANIC	Tilbrook has logged in
SIGANSI	Language too large
SIGIBM	Corporation too large
SIGAPPLE	Program is too small
ISBGTYE	Byte swap error
SIGISO	Has yet to be defined
SIGHIC	Program confused due to excess alcohol
SIGHUP	Excess curry
SIG	Attempt to execute zero length program
SIGQEII	Security violation
SIGPLAN	Unknown programming language
SIGDOS	ENOTUNIX
CIGARETTE	ENOTOBACCO
SIGCIA	Classified information
SIGPUB	Sorry, we're closing in 10 minutes
SIGLUXO	Warning -lamp enabled
SIGDAS	You've been Suniled
SIGRISC	Unimplemented instruction
SIGFREID	Beware the Valkyries
SIGCAT	No mouse
SIGCHEESE	Lost contact with mouse
SIGHIC	Bottle empty
SIGMARTINI	Received any time, any place, any where
SIGAT&T	Failed to acknowledge trademark
SIGNORTH	Security label violation
SIGYAS	Yet another signal /* Dummy signal */
SIGEEC	System deadlocked
SIGBLITZ	Program bombed out
SIG#@$:	I'm not going to tell you where the file is
SIGHOTELFULL	No more space in process table
SIGDAS	Name reference count overflow
SIGASAPARROT	Ill-eagle instruction
SIGBLUE	Unknown hardware
SIGNAL	LAN running backwards
SIGFINE	Speed trap — caught breaking 1 MIP
SIGHIC	Process beer overflow
SINGAL	Program made a spelling mistake
SIGELBOW	Wakeup (boring talk finished)
SIGCHANGLING	Child process has been replaced
SIGGURU	You are not expected to understand this
SIGBUS	Transport service failure — you have to walk home
SIGRA	Attempt to execute Pyramid code on a Sun
SIGLUXO	Your ball is flat
SIGξאψ	Message from foreign host

The Plate Competition

The second was a balloon competition. A plate had been presented as a prize by HCR, this was of the UNIX founding fathers: Dennis & Ken — but what were they saying to each other at the time that the plate was made?

The results were rather poor, however a picture of the plate and the winning caption from Bob Gray of Eucs can be found after the conference proceedings, at the back of this newsletter.

The other entries of note are:

- D'you things UNIX will fly ?
  - It will if we implement it on a frisbee

- Dennis, I've got this new operating system. I call it Unix.
  - Twenty years from now you'll regret this

- Dennis, do you think that we are famous ?
  - No Ken, we should have started in a garage

- What do you think they will eat today ?
  - Inodes I hope. They don't stick in beards

- You write the TTY driver.
  - No. You write it

The Tie Breaker

Because it can be very difficult to judge a competition with a high standard of entries (as is found at an EUUG conference ☺) it was decided to have a tiebreaker.

What happened was that the tie breaker ended up being judged as another competition. Yet another first for London: the first conference with three competitions.

Entrants were asked to complete, in no more than 20 words, the sentence:

I attend EUUG conferences because .....

Some of those replying assumed that the judges were weak, vain creatures who could be swayed by simple flattery; they claimed *because:*

The competition committee are such nice people
I think the competition judges are great
The competition judges are so intelligent
Sunil Das is my hero (crawl, crawl)
I think Sunil Das is really wonderful

I have spared the authors of the above by not printing their names, and would like to point out that Sunil was not one the judges anyway.

Various other assorted reasons given were:

- I'm a Pratt — *I'm not arguing* — ED
- I'm waiting for an engineer to arrive and have nothing better to do
- It's fun
- It has funny films
- I want to meet other hackers
- I need a good laugh
- It's the only way I've found to visit London
- My boss told be to do so
- I'll do anything to get out of the office
- Booze, views, news and reviews
- They're there

But the outstanding reason was given by Per Holck of Bull (dk):

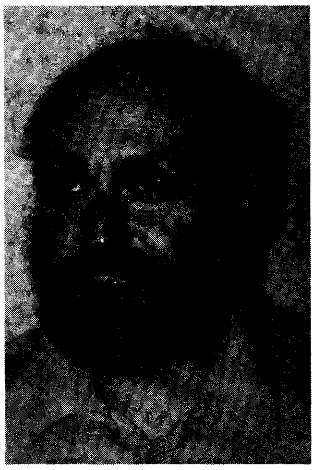
- I want to ask if anybody has received my mail

A Challenge

In the past EUUG competitions have been taken up and run by Usenix at their conferences. Opinions differ on the quality of replies obtained in the USA, they think that they are better. We disagree and assert that all the good ones are from visiting Europeans.

The EUUG hereby challenges Usenix to take up the signal competition and to try and better us at their summer conference in San Francisco.

AFUU Report — Convention UNIX '88



Philip Peake
philip@axis.uucp

Axis Digital
Boulogne
France

The most important event for the AFUU since the last article was without doubt the **Convention UNIX '88**. This was the first exhibition/conference organised directly by the AFUU, the previous exhibitions being organised by Network Events.

The preliminary information after the event gives the following statistics:

- Exhibition — 2100 square metres, with a hundred exhibitors.
- Visitors — 4700 people visited the conference, and 500 more the technical conference which was held in parallel.
- Conference and tutorials — obviously a success with that number of visitors. Since this was our first conference at a new conference centre, there were one or two organisational problems, but there will be resolved for next year.

For those interested, the proceedings of the conference are available from the AFUU.

Another important event is that the AFUU is moving. With 3 full time staff and two UNIX machines, the current office space at SUPELEC was becoming much too cramped. We are moving a little closer to Paris, in fact, just a few hundred metres from the Paris city boundary. The new address is:

AFUU
11 rue Carnot
94270 Le Kremlin-Bicetre
+33 1 46 70 95 90

This will be much easier for visitors, since it is about 50m from the nearest *metro*.

The United Kingdom UNIX Users' Group

Zdravko Podolski
Sunil K Das

sunil@nss.cs.ucl.ac.uk

Introduction

This memorandum is intended to describe the state and activities of the UKUUG. It is a modified document of that presented to the UKUUG Executive Committee, and the EUUG Governing Board and Executive Committee at the Governing Board Meeting held prior to the EUUG Spring 1988 Conference.

Membership

The membership consists of institutional, individual and honorary members. Our fees (ex VAT) are 105 pounds for the institutional membership, 50 pounds for the individual and nothing for the honorary.

The membership figure has now finally stabilised after the shocks of a couple of years ago. We would like to increase the membership and have held discussions with */usr/grp/UK* about merging which are covered later. One of the complications is that, although we recognise the need for a low cost individual membership, there needs to be control so as to avoid the tendency of everyone becoming an individual member. To encourage this, individual membership is required to be accompanied by a cheque drawn by a single person, not one drawn by a company. We look to the EUUG for help in checking at conferences etc that the person applying is a bona fide member of an institution that is an institutional member, or is an individual member in their own right. We are interested in and would support a differential charging structure, to reflect that the individual can draw much less on the services of the group than an institutional member.

The membership are very fee sensitive and we are keeping the fees at the same level as last year. We expect with a rising membership, fees can be held under reasonable control in 1989.

Meetings

There were two meetings during 1987, one at Newcastle in July and one at City University in December. The format of two half days has been successful, allowing people to travel to the meeting in the first morning, attend one session, have the evening to make and renew contacts and then have another session in the morning.

The Newcastle meeting was a straight forward technical event, with Michael Lesk as the keynote speaker, while the City event was a UKUUG workshop on computer networking in the UK attended by over 200 delegates. The workshop was closed to UKUUG members in recognition that only institutional UKUUG members are allowed to join UKNET.

The proceedings of the meetings have been published as is our normal practice, and a copy has been sent to the EUUG National Group contacts. We welcome at our events any other EUUG National Group member on the same terms as our own members. At the City meeting, for the first time and in recognition of the improving relations with */usr/grp/UK*, their members were invited to attend the meeting.

Last month, the UKUUG were hosts to the EUUG's Spring Conference held in London. We are collaborating with */usr/grp/UK* about the 'European Unix Users' Show' in London in July and holding a technical meeting in December in Cambridge. The EUUG's London meeting has consumed vast amounts of the Committee's and others' effort. The number of delegates, attendees and helpers reached a record for EUUG by touching the 600 mark.

Contacts with other groups

During the year we have kept in touch with the UK Sun Users' Group. This group is funded mostly by Sun, so it is naturally much influenced by Sun. The contacts were started with a view to some sort of collaboration, especially concerning meetings, but in the event the most we could interest them in was to avoid clashing with each others' meeting dates. We shall try harder to encourage them to cooperate with us.

We have also held several meetings with */usr/grp/UK* with a view to merging the two groups. The situation in the UK is anomalous, being the only European country with two user groups. This is an historical accident. With UKUUG/EUUG originally being seen as mostly an academic user group, the vendors started one of their own. The situation is now less clear cut, with about half the UKUUG institutional members being commercial users or vendors, while */usr/grp/UK* has been increasing the ranks of users amongst their members. We all agree on the desirability of ending this confused state of affairs and presenting a single user group to the UK UNIX users.

The structure of */usr/grp/UK* is similar to that of the UKUUG, with individual and institutional members, and fees being 200 pounds and 50 pounds respectively. However, the vast majority of their members are individual. They run a major exhibition every year (contracted out to EMAP), publish a newsletter and have various other activities and relationships, most notably with */usr/grp* in the USA.

Possibly because of their greater advertising clout they have a larger membership than us. If a merger could be achieved, the combined membership would then be in the region of 650 members. The merger discussions have reached agreement on all points except one: the EUUG fees. For a member of the new combined group, the fees would be roughly the current */usr/grp/UK* fees (50 and 200 pounds respectively) plus the EUUG 40 pounds. This would virtually double the individual members' fees overnight. There is also a problem with doubling the current UKUUG institutional fees. While savings can undoubtedly be found because of the size of the combined group, these are unlikely to be large, as both the UKUUG and */usr/grp/UK* run on a non profit basis. Various possibilities exist for solving this problem. It may be possible to reduce drastically the individual membership fee, or unbundle the various EUUG services so that those who want them just pay extra for them, or have an 'associate member' who does not get a member's reduction for conferences. We would welcome advice from the Governing Board, the EUUG executive and our membership on these points.

Another difficulty we had during our attempts to sell EUUG affiliation to */usr/grp/UK* was the lack of EUUG publicity material. The visible activities of the EUUG are well known, the conferences, the Newsletter, EUnet. However there is a whole host of efforts that can and should be loudly advertised. Links with other groups outside Europe, input to standardisation bodies, European networking initiatives, etc, etc. Some obviously should be kept quiet about until fruition, but mostly publicity would be helpful. Otherwise the EUUG is seen as a body with an expensive Newsletter and an even more expensive bureaucracy, without tangible benefits. We tried hard to disabuse them about this, and made much headway, but the issue of cost still remains unresolved.

We now need help from the EUUG to progress this initiative, which would be of immense benefit to users everywhere. The EUUG would gain many new members at a stroke while at the same time cementing the unity between users all over Europe.

USENIX Association News for EUUG Members

Donnalyn Frey
donnalyn@uunet.uucp

Fairfax
Virginia, 22031
USA

Ms. Frey is the USENIX Association Press Liaison. She provides members of the press, USENIX Association members, and EUUG members with information on the activities of the USENIX Association. Ms. Frey has been a UNIX technical writer for five years. She now writes journal and newspaper articles on UNIX-related subjects, as well as working as the Association's press liaison.

The USENIX Association has grown significantly over the last few years. Part of this growth has been the expanding contact between the members of the USENIX Association and the EUUG. This column was created to help foster this contact. The column will provide EUUG members with information on current and upcoming activities of the USENIX Association. Addresses and telephone numbers for upcoming activities are included in the column to assist EUUG members in contacting USENIX Association representatives.

Summer 1988 Conference

Most EUUG members already know about the Summer 1988 USENIX Conference and Technical Exhibition to be held in San Francisco, California on June 20—24. This is expected to be the largest USENIX Association conference to date. Several new tutorials will be presented this summer. The technical sessions include papers on subjects such as window systems, file systems, security, programming languages, and networking. Approximately 100 exhibitors will be displaying their wares at the technical exhibition. A reception will be held at the Exploratorium, a hands-on science museum in San Francisco.

The next USENIX Association Conference will be held in San Diego, California in early 1989. For information on the technical conferences or any other USENIX Association conference, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, CA 90742, USA. The telephone number is +1 213 592 1381. The email address is judy@usenix.uucp.

The Facesaver Project

The Facesaver Project will continue to record new faces and registration information at the San Francisco conference. The Facesaver combines photographs and registration information onto a sticky label that conference attendees can give to exhibitors and other attendees to help them remember each other. After the conference, the attendee list mailed to conference attendees will feature a postage stamp size portrait beside each name and address.

The Facesaver portraits are captured via a video camera using AT&T Targa M8 graphics boards installed in Bell Technologies PC AT clones running the SCO XENIX version of the UNIX operating system. Portraits are printed using a Postscript laser printer.

The Facesaver project is run by Lou Katz. It is sponsored by the Association to aid in improving attendee recognition at the conference. A sample Facesaver label is reproduced below.

FaceSaver 4/88

Donnalyn Frey
+1 703 764 9789
uunet!donnalyn
USENIX Press Liaison
P.O. Box 2051
Fairfax, VA 22031

C++ Conference

The USENIX Association is about to embark on a new series of conferences P the C++ Conferences. The Association has held successful C++ workshops in the past. However, the last workshop, in Santa Fe, New Mexico, was so well-attended that the Association decided to expand the format. The result is the first open C++ conference, scheduled for October 17—20, 1988 in Denver, Colorado. For information on technical submissions for the conference, contact Andrew Koenig at ark@europa.att.com. For registration information on this new conference, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, CA, 90742, USA. The telephone number is +1 213 592 1381. The email address is judy@usenix.uucp.

EUUG members still trying to get copies of the Proceedings of the 1987 Santa Fe, New Mexico C++ Workshop will be pleased to know that the Proceedings have been reprinted and are again available for purchase. For information on the Proceedings, contact the USENIX Association office at P.O. Box 2299, Berkeley, CA 94710, USA. The telephone number is +1 514 528 8649. The email address is office@usenix.uucp.

Large Installation System Administration II Workshop

The second Large Installation System Administration Workshop will be held November 17 & 18, 1988 in Monterey, California. The call for papers will appear in the May-June and later issues of the Association's **;login:** newsletter. For information on submissions for the workshop, contact Alix Vasilatos of MIT's Project Athena at alix@athena.mit.edu. For registration information, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, California 90742, USA, by telephone at +1 213 592 1381, or by email at judy@usenix.uucp.

UNIX Security Workshop

The UNIX Security Workshop will be held August 29 & 30, 1988 in Portland, Oregon. The workshop will bring together researchers in UNIX computer security and system administrators trying to use UNIX in environments where security is of the utmost importance.

Some topics to be included in the workshop include password security, network security, and file system security. For information on submissions for the workshop, contact Matt Bishop at bishop%bear.dartmouth.edu@relay.cs.net or at {ihnp4,decvax}!dartvax!bear!bishop. For registration information, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, California 90742, USA, by telephone at +1 213 592 1381, or by email at judy@usenix.uucp.

Workshop on UNIX and Supercomputers

The Workshop on UNIX and Supercomputers will be held September 26 & 27, 1988 in Pittsburgh, Pennsylvania. The workshop will consider the general problems of running UNIX on supercomputers and will cover both practical and abstract topics. Areas of interest will include system administration, file systems, networking, monitoring performance, shared memory management, and very large files.

For information on technical submissions for the workshop, contact Lori Grob at grob@lori.ultra.nyu.edu or Melinda Shore at shore@reason.psc.edu. For registration information, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, California 90742 USA, by telephone at +1 213 592 1381, or by email at {ucbvax,uunet}!usenix!judy.

1988—1989 USENIX Association Board of Directors

The USENIX Association recently held elections for the 1988—1990 Board of Directors. The new board is composed of:

Alan Nemeth	President
Deborah Scherrer	Vice-President
Rob Kolstad	Secretary

Michael O'Dell and Sharon Murrel are new board members. The new Board of Directors will take office at the San Francisco USENIX Association Conference in June.

2.10BSD Operating System

The USENIX Association is continuing to distribute the 2.10BSD operating system. 2.10BSD is a UNIX operating system for Digital Equipment Corporation PDP-11 series computers. It was released in October 1987 by the Computer Systems Research Group (CSRG) of the University of California at Berkeley.

2.10BSD is a basic port of 4.3BSD functionality to a PDP-11. This release is most useful for sites that have 4.3BSD programs or machines and who would like consistency across the environment. It is also useful for sites that want a faster, cleaner version of 2.9BSD.

2.10BSD is faster than 2.9BSD and includes 22-bit Qbus support; 4.3BSD networking (TCP/IP, SLIP); 4.3BSD serial line drivers; 4.3BSD C library; most 4.3BSD applications programs; 4.3BSD system calls; RAM disk; inode, core and swap caching; and the conversion of the system to a 4.3BSD structure.

This release is being handled by the USENIX Association and is available to all UNIX V7, System III, System V, and 2.9BSD licensees. The Association will continue to maintain the non-profit price of $200 plus shipping for overseas sites, as was charged by CSRG. The release consists of two 2400 foot, 1600 bpi tapes (approximately 80 MB), bug fixes, and approximately 80 pages of documentation. If a site requires 800 bpi tapes, they should contact the Association for further information.

The tapes that the USENIX Association is distributing only support machines with split I/D and floating point hardware. At this time, the Interlan Ethernet, DEQNA, DEUNA,and 3COM drivers have been ported. 2.10BSD runs on the following machines: PDP-11 /24 /34 /44 /53 /60 /70 /83 /84 and PDP-11/23

For information on the distribution of the 2.10BSD release, contact the USENIX Association 2.10BSD, P.O. Box 2299, Berkeley, CA 94710, USA. The telephone number is +1 415 528 8649. The email address is office@usenix.uucp.

More to Come...

This column will again appear in the Autumn EUUG newsletter, providing EUUG members with more news of the USENIX Association. Any (favourable) comments on the column should be sent to me.

Computing for the 1990's

Malcolm Stayner
malcolm@nezsidc.uucp

ICL New Zealand Ltd
Wellington
New Zealand

The forthcoming NZUSUGI (New Zealand UNIX Systems User Group Inc.) Conference and Exhibition at the Plaza International Hotel, Wellington on 9—11th June promises to be the biggest and best yet. With the sale of several large UNIX systems marking the very definite arrival of UNIX as an operating system for commercial installations in New Zealand, interest from vendors has been very high. All the exhibition space was sold by early March and visitors to the exhibition can expect to see some exciting developments in the UNIX world. The deluxe exhibitors will be ICL, NCR, NEC and Olivetti who will, no doubt, be exhibiting their products to the utmost. Premium stand holders are Altos, Phoenix, Unisys and STC. Finally, also exhibiting will be Apple, Barson, CBA, Northrop, Rakon, Today and Xact. The exhibition will be open to the public on all three days.

'Computing for the 1990's' is the theme for the conference, chosen because UNIX is seen as providing the vehicle for departmental computing, a major growth area for the next decade. It is perhaps worth pointing out that it is not envisaged that departmental systems will oust PCs or mainframes but rather form a worthy complement to these well established areas of information processing. While the individual will still require computing power on his or her desk and corporations will still require centralised data storage and processing, there is a large middle ground where information has to be shared amongst a work group. It is here that UNIX-based systems are seen as providing a solution to a pressing business need, which is the requirement for two or more people with a common business function (e.g., scheduling sales calls or ordering supplies) to be able to communicate and operate effectively, thereby increasing their contribution to the organisation. As such, topics such as standards and connectivity are major issues and the conference will address these. This year the conference will specifically target government departments as well as our traditional audience, as NZUSUGI believes that there is much scope for UNIX-based systems amongst this community.

We are honoured to have Peter Dunne, Under Secretary for Trade and Industry address the conference on the opening day and formally open the exhibition.

Once again, there is an excellent line-up of international and local speakers. From overseas there is Stuart Hooper, Director of The Santa Cruz Operation, who will speak on the significance of the merging of UNIX and XENIX; Jim Bennett of Convergent Technology will speak on Windows under UNIX and Network File Sharing and James Clark of the Singapore UNIX Association will speak on the Universality of UNIX in the Asian culture.

Our cast of overseas speakers will be complemented by ten local speakers addressing various UNIX-related issues, including electronic mail, office automation and UNIX trends in government worldwide. Doug Marker of IBM will talk about IBM's UNIX strategy and the differences between AIX and OS/2. Keith hopper, a NZUSUGI stalwart, will talk on POSIX, Mark III and beyond.

We are delighted to have as chairpersons for our stream sessions this year Ian Howard and Roger Hicks, co-directors of Rakon Independent Software Division, and John Hine from the Department of Computer Science of Victoria University, Wellington.

Finally, the conference will be brought to a close by what promises to be a very lively debate among a panel of vendors and users. The topic for discussion will be 'Standards, are they effective?' This will be chaired by Gordon Hogg, former Chief Executive of Databank.

The annual NZUSUGI conference provides a valuable opportunity for NZUSUGI members and those involved with UNIX to mix and mingle in a convivial atmosphere. The conference dinner on the evening of 9th June will be one of the social highlights. Mike Dubrall, Director of the Independent Software Marketing Division for the Pacific Group of NCR, who has attended two past NZUSUGI conferences said in UNIX/World (Oct'87) that the NZUSUGI conference was "arguably one of the best UNIX gatherings in the world today". He went on to say, "all in all, the New Zealand UNIX Group is as vibrant as any in the world. UNIX aficionados would do themselves a favour by combining a vacation with the business of attending the next NZUSUGI conference in Wellington". Such an accolade from the United States is a fearsome one indeed, and one that this year's Conference Committee will be endeavouring to surpass.

Malcolm Stayner
1988 NZUSUGI Conference Chairman

For further details contact:

Malcolm Stayner
Technical Manager, Software Industry Development Centre
ICL New Zealand Ltd
P.O. Box 394
Wellington
New Zealand

Tel: +64 4 724-884
FAX: +64 4 726-737

For a registration form contact:

Jan Tonkin
CMS
P.O.Box 12-442
Auckland

Tel: +64 9 525 1240
FAX: +64 9 525 1243

CALL FOR PAPERS AUUG '88

Australian UNIX® systems User Group Winter Conference and Exhibition 1988 September 13 – 15 1988, Melbourne, Australia

Summary

The 1988 Winter Conference and Exhibition of the Australian UNIX systems User Group will be held on Tuesday 13th – Thursday 15th September 1988 at the Southern Cross Hotel in Melbourne, Australia. The conference theme is

Networking – Linking the UNIX World.

AUUG is pleased to announce that the guest speakers will include:

Ken Thompson	Bell Laboratories
Michael Lesk	Bell Communications Research
Mike Karels	University of California at Berkeley

Papers

Papers on topics related to computer networks and UNIX, and on non-networking aspects of the RNIX system are now invited. Some suggested topics include but are not restricted to:

- Operating system and programming language support for networks
- Distributed file systems and their application
- Networked window systems
- ISO/OSI and UNIX
- Security aspects of computer networks
- Legal and social aspects of computer networks
- Protocol specification methods
- Harnessing new technologies
- Network applications under UNIX

Papers on other (non-networking) aspects of the UNIX system are also sought.

Authors of papers presented at the conference will receive complimentary admission to the conference and the dinner. AUUG will again hold a competition for the best paper by a full time student at an Australian educational institution. The prize for this competition will be an expense paid return trip from within Australia to the conference to present the winning paper. A cash prize in lieu of this may be paid at the discretion of AUUG. Students should indicate with their abstract whether they wish to enter the competition. AUUG reserves the right to not award the prize if no entries of a suitable standard are forthcoming.

A special issue of the group's newsletter AUUGN containing the conference proceedings will be printed for distribution to attendees at the conference.

EUUG

European UNIX® systems User Group

Autumn 1988 CONFERENCE

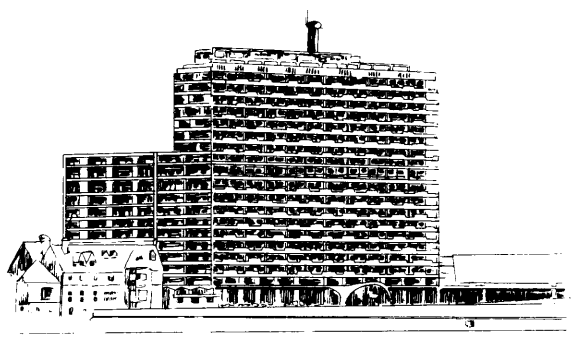
Tutorials and Showcase

New Directions for UNIX

Hotel Estoril-Sol

Cascais, Portugal

3-7 October 1988



The **EUUG Autumn '88 Conference** will be a truly European event giving a Forum for the varied and interesting work on UNIX which is currently taking place in Europe. All the lecturers will be recognised authorities on the various fields to be discussed.

The main conference is expected to cover such areas as **Real Time; Security Issues; Distributed Processing; Multi-processors and Parallelism; Supercomputing; Internationalisation; Fault Tolerance; Transaction Processing; Virtual Memory; Object Oriented Approaches; Videotext Applications; Standards and Conformance Tests.**

The conference will be complemented by two days of Tutorials, on 3rd and 4th October, which will cover the following subjects:
Users introduction to the X Window System; POSIX Implementation; and Curses (Day 1) plus Programming with the X Window System; Sendmail; and System Performance & Management (Day 2).

For further details on this important event, including information on accommodation, please contact:

European UNIX® systems User Group

Owles Hall, Buntingford, Herts SG9 9PL, UK
Tel: Royston (0763) 73039 +44 763 73039 (overseas)
Facs: Royston (0763) 73255 +44 763 73255 (overseas)
Network address: euug@inset.uucp

Cost of attending the conference will be £240 for members of EUUG and £280 for non-members if booked and paid for before 22nd August 1988. Cost does not include accommodation which must be booked and paid for separately, but it does include coffee, tea, lunch, a copy of the proceedings and attendance at the gala folk evening dinner.

Tutorials

Tutorials will be held on 3 & 4 October 1988 but will be open to members only. The cost will be £176 for one whole day tutorial including refreshments, lunch and course notes if booked and paid for before 22nd August 1988.

(Each tutorial lasts for one whole day and will start at 09.30)

The Tutorials will be:

Monday 3rd	*Users introduction to the X Window System*	*POSIX Implementation*	*Curses*
Tuesday 4th	*Programming with the X Window System*	*Sendmail*	*System Performance & Management*

In outline they will cover:

M1
Users Introduction to the X Window System — *By Watson, Clenton & Caruthers*

This tutorial is intended for those who will be using the X Window System, and for managers of such people. It is NOT a programming tutorial of any kind; there will be no discussion of programming, libraries, toolkits, or other such topics. It may be of interest, however, to those who will be programming with the X Window System, but as yet have never worked with it. No prior experience with windowing systems is necessary or assumed. Topics covered will include:

- General concepts of windowing systems
- Networking concepts, Network transparency in X
- Structural overview of the X Window System
- Display server concepts; windows, icons, mouse, keyboard
- Window manager concepts, interacting with window managers, menus
- Two significant window managers; uwm and menuwm
- Configuring and customizing X; .Xdefaults, etc

M2
POSIX Implementation — *By Shane P. McCarron & John Quarterman*

This tutorial is about the IEEE 1003.1 portable operating interface standard, POSIX. The tutorial starts with a brief history of the standard and its relations to other standards, such as the 1984 /usr/group standard, the X/OPEN Portability Guide, and the ANSI X3.159 C Standard. It ends with implications of related POSIX draft standards, such as 1003.2 (shell necessary to implement applications that will run on a POSIX conforming implementation and be as portable as possible to other such interfaces. It includes suggestions on what to do and what to avoid in a number of troublesome areas, such as signals, pipes and FIFOs, and the terminal interface.

Attention is paid to the differences between the POSIX interface and those of System V Release 3 and 4.3 BSD. This tutorial is primarily oriented toward applications developers who want an understanding of the portability restrictions required by POSIX, and how those restrictions relate to their current programming practices. However, since the session is a summary of the differences between POSIX and current implementations, it will also be of use to system implementors who need to understand what enhancements are required for POSIX conformance.

Attendees should be experienced application writers who want to learn how to take best advantage of POSIX in writing portable programs, and interface implementors who want to know what application developers expect in their interface. Attendees should have read the POSIX standard beforehand. There is no license requirement for attendance.

Shane P. McCarron is a Systems Analyst with the Minnesota Educational Computing Corporation. He joined the 1003.1 working group in 1985, and became the secretary in September, 1986. In addition to his role in P1003.1, he is also a member of P1003.2 and ANSI X3J11.

M3
Curses — *By Prof. Axel Schreiner*

Introduction:	What are curses? What is terminfo? NOT: How do I write terminfo for a new terminal?
Curses-Desk-Calculator:	a program where all curses functions can be tried out interactively — participants are welcome to sources Effects of the various functions (families) are shown using this program.
Window-Shell:	a program which manages several windows on one terminal screen and permits switching between them — participants are welcome to sources. Windows can be opened and closed, moved, and resized. Inside a window, one can use a C desk calculator, look at files or command output, run an interactive program such as a shell, full-screen edit, a file, or look at on-line help. The window-shell illustrates very many typical problem settings for screen-oriented applications and offers solutions that can easily be extracted. All curses functions dealing with a single terminal are used in the window-shell.

The tutorial will concentrate on building the window-shell and dealing with all curses — and system-call-related problems. Knowledge of C is essential, knowledge of lex and yacc would be helpful but is not mandatory.

T4
Programming with the X Window System — *By Watson, Clenton & Caruthers*

This tutorial is intended for programmers who will be developing applications or utilities based on the X Window System. It is assumed that attende[illegible]ll be competent programmers, and will have some fundamental knowledge of windowing systems in general, and preferably of the X Window System in particular. Those without such experience would benefit greatly from attending the previous class before attending this one. Topics covered will include:

— *Basic programming concepts of the X Window System*
- Coordinate systems
- The byte stream data connection
- Conceptual objects used in X
- X data types, structures

— *Xlib interface*
- Naming conventions, argument passing and return
- Detailed examination of common routines

— *Toolkits*
- Xtk, the "standard" toolkit
- Xray, a very popular alternative
- InterViews, object oriented toolkit in C + +

T5
Sendmail — *By Forrest Smoot Carl — Mitchell, Managing Partner of Texas Internet Consulting*

I. Overview

A. Function: 1. Routing; 2. Address Rewriting; 3. Queuing; 4. Mail delivery; 5. Aliasing.

B. Architecture: 1. Sendmail program; a. Queue daemon; b. SMTP daemon; c. Deliverer.

2. Support files and directories: a. sendmail.cf & sendmail.fc; b. mail queue; c. aliases; d. sendmail.hf; e. sendmail.st; f. booting; g. forwarding mail

3. Support programs and functions: a. new aliases; b. mailq; c. syslog

4. Typical mail deliverers: a. /bin/mail; b. /usr/bin/uux; c. SMTP delivery

C. Typical installations: 1. Stand alone; 2. Simple UUCP site; 3. Simple ARPA Internet site; 4. Complex mail relay

II. Configuration file
A. Format and syntex; B. Boilerplate stuff; C. Address and header rewriting: 1. How rewriting works; 2. The macro language: a. Special macros and classes; b. User defined macros and classes; 3. Rewriting rules and rulesets; 4. Special rulesets: a. 0 (zero) — find the right mailer; b. 3 — canonical form; c. 1, 2 & 4 — header rewriting; 5. Mailer configurations: a..local mailer; b. SMTP mailer; c. UUCP mailer

III. Hints and tidbits
A. Sendmail and the ARPA Internet namesaver: 1. Canonical names and aliasing problems; 2. Being sure headers are correct; B. Cluster configurations: 1. Workstation cluster; 2. General addressing and making all machines local; 3. Sharing aliases in a LAN environment; C. Setting up mail relays and gateways: 1. Small sites; 2. Single host; 3. Large complex sites; D. Tuning hints: 1. Queueing versus direct delivery; 2. Delivery modes; 3. File protections. E. mailer options and which ones are useful; F. Debugging hints: 1. using -v; 2. using -d; 3. testing configurations; 4. finding the value of a built-in macro; G. Special headers and aliases: 1. MAILER-DAEMON; 2. Apparently-from; 3 Received from lines and debugging

T6
System Performance & Management —
By Brian E.J. Clark, Pyramid Technology

This tutorial concentrates on two areas of interest to the managers of large UNIX based systems.

As UNIX gains credibility as a "standard" delivery vehicle for a wide range of different applications environments, the ability to assess and manage the systems performance has become increasingly important.

This tutorial deals with both performance measurement and systems management. It will carry out a full review of the performance measurement tools available as part of the normal UNIX tool set, and also examine the products becoming available within the commercial marketplace.

Some time will be spent examining the output from these tools and examining how they relate to the load on the running system and its hardware configuration.

As the power of systems increases, so do the problems of system management. Terms such as mainframe and supermini become increasingly confused as superminis find their way into mainframe environments.

UNIX alone offers very little in the way of systems management tools, relying on the skill of the System Administrator to devise the necessary tools out of the UNIX toolkit.

As UNIX permeates the commercial marketplace this approach becomes less viable and so there is a range of tools emerging to answer this need. The second part of this tutorial will concentrate on these tools, why they are needed, and how they can be used to provide the management facilities needed for the commercial systems manager.

Accommodation
Accommodation may be booked direct with the five star ESTORIL-SOL, where the conference is being held, by using the booking slip provided. Price for each person is as follows:

Please note that the hotel requires a deposit at the time of booking.

HOTEL ESTORIL SOL

Double room each person per day	£37
Single room each person per day	£65

The Secretariat can also put delegates in touch with hotels offering cheap accommodation and can suggest alternative accommodation once the above hotel is fully booked. Early booking is recommended to avoid disappointment.

Cancellations
It is regretted that no refund of fees will be possible in case of cancellation, unless the cancellation is made more than one month before the start of the Conference. No cancellation will be accepted unless it is sent to the EUUG Secretariat in writing.

Language
The official language of the Conference will be English. No translation will be provided.

Liability
The EUUG will not accept any responsibility for damage to property or injury to persons during the entire event. Participants are recommended to arrange for their own personal travel and health insurances.

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

Applications should be made on the form at the back of this booklet, together with a booking for the Conference on the other form. If your booking is dependent on obtaining the grant, please write on the top of your booking form: **"GRANT DEPENDENT".**

Priority will be given to:

1. Students giving a talk at the Conference

2. Students doing work for the EUUG or a National Group

3. Students

4. Other deserving cases like research students.

You can apply for (partial) coverage of expenses for travel in Europe, accommodation and Conference fees, but not for meals. Student status or other deserving status, must also be documented by a copy of a valid student registration card or the like. After the event, original bills must be included with the claims form. The Secretariat can help you find suitable student accommodation.

Exhibition
A showcase Exhibition will be open during the event. Companies wishing to exhibit should contact the EUUG Secretariat for full details and booking form.

How to Book
To book a place at the Tutorials and/or Conference, complete one booking form for each person and return it *with* the full remittance to:

The Secretariat	tel:	+44 763 73039
European UNIX systems User Group	fax:	+44 763 73255
Owles Hall, Buntingford	email:	euug@inset.uucp
Hertfordshire SG9 9PL, UK	or	...!mcvax!ukc!inset!euug

Use a photocopy of the booking form for each additional person. Please note that bookings can only be accepted when accompanied by payment. Telephone bookings will not be accepted. The EUUG Secretariat will acknowledge your booking by sending you a receipted invoice together with further details for registration.

All payments must be made in pounds sterling (£).

Payments may be made in one of 3 ways.

1. By UK Cheque or Bankers Draft, made payable to EUUG, and drawn on a UK bank. Eurocheques are acceptable, but each cheque must be £100 or less.

2. By Direct Payment to the EUUG's bank, which is:

The Bank of Scotland	Account Number: 00613997
61 Grassmarket	Bank Sorting Code: 80-31-50
Edinburgh	
Scotland EH1 2JF	

Please tell your bank that you will pay all charges so that EUUG will receive the full amount due.

3. By VISA, with card details appearing on the Booking Form.

NOTE: Closing date for all bookings is 26th September 1988.

Costs

		Cost if booked and paid for *before 22nd August*	*after 22nd August*
EUUG national group or direct members			
3 day conference	1 person	£240	£260
1 day tutorial	1 person	£176	£195
Non-members			
3 day conference	1 person	£280*	£300

* *The reduced fee also applies if 3 or more delegates apply together.*

Booking Form for Conference and Tutorials

Please complete this form and send it, with cheque or evidence of payment, to **EUUG Secretariat, Owles Hall, Buntingford, Herts SG9 9PL, UK** (Block capitals please)

Surname ______________________ **Usual first name** ______________________

Company/Organisation ______________________

Address ______________________

____

Country ______________________ **Post/Zip Code** ______________________

Telephone/Fax/Telex/Email ______________________

EUUG member? Yes ☐ No ☐ Student? Yes ☐ No ☐

Please read the sections on **"COSTS"** and remember that pre-booking saves money.
All payments must be made in pounds sterling (£).

CONFERENCE
Please reserve me a 3 day place for the Technical Sessions. ... ... £ ______

TUTORIALS (members only)
Please reserve me a place for Tutorial No______on Monday, 3rd October ... ... £ ______
Please reserve me a place for Tutorial No______on Tuesday, 4th October ... ... £ ______

Do you require Vegetarian meals Yes ☐ No ☐

TOTAL £ ______

EUUG
Please enrol me as an institutional member of EUUG via the appropriate national group Yes ☐ No ☐

Please read the section **"HOW TO BOOK"**

PAYMENT METHOD

☐ UK Cheque, Bankers Draft or Eurocheque. The cheque must be enclosed.

☐ Direct Payment. The bank advice note showing details and date of payment must be enclosed. *All bank charges must be borne by you and not the EUUG — please tell the bank this.*

☐ by **VISA**

Name as it appears on the card (block capitals) ______________________

Address of cardholder ______________________

____

____

Card Account No. ______________________ Date of Expiry ______________________

Signed ______________________ Date ______________________

Student Grant Application Form

Return to **EUUG Secretariat, Owles Hall, Buntingford, Herts SG9 9PL, UK**

(Block Capitals Please)
Name ....

Address ....

....

....

....

Telephone ....

EUnet address ....

Position ....

University/Organisation ....

Status Speaker: ☐ Helper: ☐ Student: ☐ Other: ☐ tick as appropriate

Student status or other deserving status must be documented by a copy of a valid student registration card or the like.

Expenses requested/reclaimed (Give best estimate where necessary):

Travel ....

Accommodation ....

Conference fee ....

Total amount ....

Date ....

Signature ....

OFFICE USE ONLY
Received EUUG .... Granted .... Amount ....

PRODUCT ANNOUNCEMENT

Miranda - release one

Miranda™ is a new functional programming language designed by Professor David Turner of the University of Kent. It is based on the earlier languages SASL, KRC, and ML. A program in Miranda is a set of equations describing the functions and data structures which the user wishes to compute. A program written in Miranda is typically 10 to 20 times shorter than the corresponding program in C or PASCAL. The main features of Miranda are:

* Purely functional - no side effects
* Higher order - functions as values
* Fully lazy - infinite data structures
* Concise notation for sequences
* Polymorphic strong typing

The basic types of the language are numbers (integer and double precision floating point), characters, booleans, lists, tuples, and functions. In addition a rich variety of user-defined types may be introduced by writing appropriate equations. There is a mechanism for abstract data types, and a simple but powerful module system with separate compilation and full type security across module boundaries. A more detailed discussion of the language can be found in *"Miranda: a non-strict functional language with polymorphic types"*, in Springer Lecture Notes in Computer Science, vol 201.

The Miranda system provides an interactive programming environment, running under UNIX™. The Miranda compiler works in conjunction with a screen editor (normally this is **vi**, but it is easy to arrange for this to be another editor if required). Programs are automatically recompiled after edits and any syntax or type errors signalled immediately. The polymorphic type system enables a high proportion of programming errors to be trapped at compile time. There is an online reference manual, which documents all aspects of the Miranda system. Execution is by a fast interpreter, using an intermediate code based on combinatory logic. There is a built-in 'make' feature which keeps object files up-to-date with their sources.

release information

The Miranda™ system has been developed by Research Software Ltd, a company formed in 1983 to develop and bring to market advanced functional programming systems. The Miranda system has been extensively tested and is now running at over 200 sites. Release one is a major new release incorporating separate compilation, numerous enhancements and a revised version of the Miranda language.

The Miranda system is currently available for the following machines - **VAX/microVAX** under Berkeley UNIX or ULTRIX, **SUN 2**, **SUN 3** and **Apollo** workstations, **GOULD** and **ORION** superminicomputers, **AT&T 3B series** (3B2 and upwards) under system V. Ports to several other UNIX based machines are planned in the near future.

The license fee, per cpu, is £400 for an educational license and £1400 for a commercial license. (US prices $640 educational, $2450 commercial.) These prices are valid to 31 January 1989. Network licensing terms available on request.

To find out more about the Miranda system, fill in the following form and mail it to:

Research Software Ltd
23, St Augustines Road
Canterbury, Kent CT1 1XP
ENGLAND
telephone (+44) 227-471844
email *mcvax!ukc!mira-request*

I may be interested in obtaining a copy of the Miranda system - please send further information.

Name:
Position:
Organisation:
Address:

Machine type:

Miranda is a trademark of Research Software Ltd.
UNIX is a trademark of AT&T.

Acceptance of papers will be based on an extended abstract and will be subject to receipt of the final paper by the due date.

The format of final papers will be specified with the letter of acceptance.

Abstracts and final papers should be submitted to the programme committee chair:

Tim Roper	Phone:	International	+61 3 5871444
AUUG 88		National:	03 5871444
Labtam Limited	Fax:	International:	+61 3 5805581
PO Box 297		National:	03 5805581
Mordialloc	Telex:	LABTAM AA33550	
Victoria 3195	ACSnet:	timr@labtam.oz	
Australia	UUCP:	uunet!munnari!labtam.oz!timr	
	ARPA:	timr%labtam.oz@uunet.uu.net	

Final papers may be sent via electronic mail and formatted using *troff* and any of the standard UNIX macro and preprocessor packages (–ms, –me, –mm, pic, tbl, eqn) or with TeX or LaTeX. Alternatively, final papers may be submitted as camera ready copy on A4 paper with 35mm margins left at the top and bottom. Intending authors unable to produce either of these forms are requested to contact the programme committee chair.

Timetable

Receipt of Extended Abstracts	Monday 13th June
Letters of Acceptance Sent	Monday 4th July
Receipt of Final Papers	Monday 8th August
Conference and Exhibition	13th–15th September

Ten Years Ago ...

At the DECUS conference held at Bath University the UNIX Sig held its first technical meeting. Amongst others Peter Gray, Alistair Kilgour, Emrys Jones and Zdravko Podolski gave papers.

Hot topics of discussion:

The 'new' VAX11/780
Putting frequently used commands in /bin and rest in /usr/bin
A program to rearrange files on disk to avoid fragmentation
The Glasgow tty driver

Conference dinner in the Pump Room, after a civic reception in the old Roman Baths. The water was truly awful, but the dinner was great. Maybe we should consider it for an EUUG/UKUUG meeting?

Thank you for this, Zdravko. Does anybody else have long memories that they would be willing to share with the rest of us? Please mail items to the editor.

European UNIX® systems User Group
Owles Hall, Buntingford, Herts. SG9 9PL, UK
Tel (+44) 763 73039

PRELIMINARY ANNOUNCEMENT

and

CALL FOR PAPERS

–=–=–=–=–=–

EUUG SPRING '89 CONFERENCE

Brussels, 10–14 April 1989

–=–=–=–=–=–

UNIX: EUROPEAN CHALLENGES

Preliminary Announcement

The BUUG will host the Spring '89 European UNIX systems User Group Technical Conference in Brussels, Belgium.

Technical tutorials on UNIX and closely related subjects will be held on Monday 10th and Tuesday 11th April, followed by the three day conference with commercial exhibition finishing on Friday 14th April.

A pre-conference registration pack containing detailed information will be issued in early December 1988.

Call for Papers

The EUUG invite abstracts from those wishing to present their work. All submitted papers will be refereed to be judged with respect to their quality, originality and relevance.

Suggested subject areas include:

- Real time
- Networking
- Security issues
- Graphics
- Internationalisation
- Distributed processing
- Fault tolerance
- New architectures
- Transaction processing
- Window systems and environments
- Supercomputing
- Standards and conformance tests

Submissions from students are particularly encouraged under the EUUG Student Encouragement Scheme, details of which are available from the EUUG Secretariat.

Important Dates

Abstract deadline	**November 30th, 1988**
Acceptance notification	**January 15th, 1989**
Final paper received	**February 1st, 1989**

Method of submission

Abstracts **must** be submitted by post to the EUUG Secretariat. All submissions will be acknowledged by return of post.

Papers may be submitted electronically to Prof Nyssen, but this is not the formal method of submission.

Tutorial Solicitation

Tutorials are an important part of the EUUG's biannual events providing detailed coverage of a number of topics. Past tutorials have been taught by leading experts. Those interested in offering a tutorial should contact the EUUG Tutorial Officer as soon as possible.

Additional Information

The Programme Chair, Prof Marc Nyssen, will be pleased to provide advice to potential speakers.
Prof Nyssen may be contacted at the address below.

If you wish to receive a personal copy of any further information about this, and future EUUG events, please write, or send electronic mail, to the Secretariat.

Useful Addresses

Secretariat	Tutorial Officer	Programme Chair
EUUG	Neil Todd	Prof Marc Nyssen
Owles Hall	IST	Medical Informaticas Dept
Owles Lane	60 Albert Court	Vrije Universiteit Brussel
BUNTINGFORD	Prince Consort Road	Laarbeeklaan 103
Herts	LONDON	B-1090 JETTE
SG9 9PL	SW7 2BH	B-1090
UK	UK	BELGIUM
Phone: (+44) 763 73039	(+44) 1 581 8155	(+32) 2 477 44 24
Fax: (+44) 763 73255	(+44) 1 581 5147	(+32) 2 477 40 00
Telex:	928476 ISTECH G	
Email: euug@inset.uucp	neil@ist.co.uk	marc@minf.vub.uucp

Macro Expansion as Defined by the ANSI/ISO C Draft Standard

Cornelia Boldyreff
Cornelia.Boldyreff@brunel.ac.uk

PRACTITIONER PROJECT
Department of Computer Science
Brunel University

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

Introduction

The ANSI/ISO C Draft Standard contains a definition of the C Preprocessing Directives in Section 3.8. As the standard is not a tutorial, it simply aims to provide a definitive guide for implementors and programmers. Occasionally examples are included in the text of the standard as illustrations; however, the non-tutorial character of the document precludes any detailed explanation of these examples. In this note, the examples given to illustrate aspects of macro expansion are discussed in more detail.

This note arose out of explanations prepared in response to comments submitted to ISO purporting to have found an inconsistency and an error in the preprocessing examples. Examples 1 and 1a: Illustrations of macro-replaced `#include` directives

Example 1 found in section 3.8.2 is unlikely to surprise current users of C; under the proposed ANSI/ISO C standard, the preprocessing tokens following the `#include` are subject to processing, that is any macro names found will be subject to replacement. The result must be a header name as defined in section 3.1.7; basically, a header name is a sequence of characters delimited by either angle brackets `(<>)` or double quotes with some restrictions on allowable characters.

In the example given, one of a series of macro definitions is selected on the basis of a nested set of conditional inclusions. This macro name following the `#include` is then replaced by the replacement list in the selected definition; and the result forms a valid header name as required.

```
#if VERSION == 1
   #define INCFILE "vers1.h"
#elif VERSION == 2
   #define INCFILE "vers2.h"
#else
   #define INCFILE "versN.h"
#endif
#include INCFILE
     /* Example 1 */
```

An example with a similar effect can be extracted from those given in the draft to illustrate rules for creating character string literals and concatenating tokens; it is given below.

```
#define str(s) # s
#define xstr(s) str(s)
#define INCFILE(n) vers ## n
#include xstr(INCFILE(2).h)
     /* Example 1a */
```

This example works because an argument may consist of any number of (but at least one) preprocessor tokens; and any argument not corresponding to a parameter that is an operand of either the `#` operator or the `##` operator is subject to complete macro replacement before substitution. By invoking the macro `str` through the macro `xstr`, this example ensures that any macros in the argument of `str` have been completely replaced.

A similar effect may achieved by explicitly invoking a `paste` macro either directly or through another macro, e.g.

```
#define paste(a,b) a ## b
#define xpaste(a,b) paste(a,b)
#define str(s) # s
#define xstr(s) str(s)
#include xstr(xpaste(paste(vers, 2), .h))
```

This explicit form is more general as Example 1a only works where the macro invocation can be clearly distinguished in the argument. The following would not succeed:

```
#define str(s) # s
#define xstr(s) str(s)
#define INCFILE vers2
#include xstr(INCFILEheader)
```

Example 2: Illustrations of Redefinition and Re-examination

This example is best considered as several smaller examples. The directives given define a series of macros; these are listed below:

```
#define x 3
#define f(a) f(x * (a))
#undef x
#define x 2
#define g f
#define z z[0]
#define h g(~
#define m(a) a(w)
#define w 0,1
#define t(a) a
```

The scope of the initial definition of `x` [line 1] extends to the corresponding `#undef` directive [line 3]; as there are no relevant occurrences of `x` in this section of the file, these two lines may be ignored. The expression given in the text:

```
f(y+1) + f(f(z)) % t(t(g)(0) + t)(1);
g(x+(3,4)-w) | h 5) & m
      (f)^m(m);
```

yields the result:

```
f(2 * (y+1)) + f(2 * (f(2 * (z[0])))) % f(2 * (0)) + t(1);
f(2 * (2+(3,4)-0,1)) | f(2 * (~ 5)) & f(2 * (0,1))^m(0,1);
```

To clarify how this result is obtained, consider the expansion of the following terms:

Number	Term	Result
1	`f(y+1)`	f(2 * (y+1))
2	`f(f(z))`	f(2 * (f(2 * (z[0]))))
3	`t(t(g)(0) + t)`	f(2 * (0)) + t
4	`g`	f(2 * (2+(3,4)-0,1))
5	`h`	f(2 * (~5))
6	`m(f)`	f(2 * (0,1))
7	`m(m)`	m(0,1)

The two key phrases to note are:

— Before substitution, each argument's preprocessing tokens are completely macro replaced as if they formed the rest of the source file; no other preprocessing tokens are available. [section 3.8.3.1]

and

— [Under 3.8.3.4 Rescanning and further replacement] If the name of the macro being replaced is found during this scan of the replacement list [i.e., the scan after substitution] (not including ...), it is not replaced. Further, if any nested replacements encounter the name of the macro being replaced, it is not replaced. These non-replaced macro name preprocessing tokens are no longer available for further replacement ... [This effectively rules out recursion. See note in Rationale.]

The processing of term 1 is simply an expansion of macro `f` with argument `y+1`. As the argument contains no macros, after substitution `f(y+1)` becomes: `f(x * (y+1))` and `x` is replaced by `2` on the rescan giving: `f(2 * (y+1))`.

The expansion of macro `f` in term 2 is as follows: Parameter a matches argument `f(z)`. Before substitution, `f(z)` is expanded with a matching z; and then `z` in turn is expanded to `z[0]` and this `z` is not subject to any further replacement. Thus after substitution, `f(z)` becomes: `f(x * (z[0]))` and `x` is replaced by `2` on the rescan giving: `f(2 * (z[0]))`. So the original expression `f(f(z))` becomes: `f(x * (f(2 * (z[0]))))` after substitution. Macro `x` is replaced by `2` on rescan giving the following as the final result: `f(2 * (f(2 * (z[0]))))`.

The expansion of term 3 is as follows: parameter `a` matches argument `t(g)(0) + t` which requires further expansion of `t(g)` before substitution. So `t(g)` is expanded with parameter `a` matching `g`; and `g` is further expanded to `f`; so on substitution, `t(g)` becomes `f`. On substitution in the original expression, the replacement list becomes `f(0) + t` which on rescan results in expansion of `f(0)` and no expansion of `t` as it is ruled out by section 3.8.3.4. The subexpression `f(0)` expands to `f(x * (0))` and with `x` being replaced by 2 becomes `f(2 * (0))`. Thus, the final result is `f(2 * (0)) + t`.

Term 4 is simply a replacement of the macro `g` by the replacement list `f`. This token is followed by a left parenthesis so the following tokens are interpreted as the argument to an invocation of the macro `f`. The result of invoking `f` with the argument `x+(3,4)-w` is given by replacing the `x` and `w` macros, and then substituting the result in `f`. The argument before substitution is `2+(3,4)-0,1` and after substitution and replacement of `x` with 2, the final result is: `f(2 * (2+(3,4)-0,1))`.

In term 5, `h` is replaced by `g(~`; on rescan `g` is replaced by `f` and on the next rescan `f` followed by a left parenthesis is recognised as an invocation of the macro `f` with argument `~5`. After substitution and replacement of `x` with, the result is: `f(2 * (~5))`.

The sixth term, `m(f)`, is an invocation of macro `m`; after substitution, the result is: `f(w)`. Note the intervening newline in the source is simply interpreted as white space which is lost during replacement. On the rescan, this recognised as an invocation of `f` with argument `w` which expands to `0,1` before substitution; and `f(x * (0,1))` after substitution. After replacement of `x` by `2`, the final result is: `f(2 * (0,1))`.

The last term is quite straight forward to explain; this is an invocation of macro `m` with argument `m` matching parameter `a`. After substitution, the result is: `m(w)`. On rescan, the `m` is not subject to further replacement by the rule given in 3.8.3.4; and the `w` is replaced by `0,1` giving the final result: `m(0,1)`.

Examples 3, 4 and 5: Illustration of rules for creating character string literals and concatenating tokens

The series of definitions listed here can be separated into four separate examples; one has already been discussed as 1a above. The relevant definition for example 3 are:

```
#define debug(s, t) printf("x" # s "= %d, x" # t "= %s", \
                    x ## s, x ## t)
```

and the term to be processed consists of: `debug(1, 2)`. After substitution, the result is: `printf("x" "1" "=%d, x" "2" "=%s", x1, x2)`.

Example 4 illustrates the special handling required for producing the spelling of string literals and character constants. The relevant definition is:

```
#define str(s) # s
```

and the terms to be expanded are:

```
str(strcmp("abc\0d", "abc", '\4') == 0)
str(: @\n)
```

Only in the first case does the argument receive special handling to deal with string literals and character constants; in the latter case, there are none. Thus `str(:@\n)` simply expands to `": @\n"`. In the first case, a backslash is placed before each double quote and \ character of a character constant or string literal before this argument is transformed into a single character string literal which replaces the list `# s`. Thus, the result after replacement is:

```
"strcmp(\"abc\\0d\", \"abc\", '\\4' == 0)"
```

The fifth example simply illustrates the difference between invoking the `paste` operation directly and indirectly. The relevant definitions are:

```
#define glue(a, b)  a ## b
#define xglue(a, b)  glue(a, b)
#define HIGHLOW  "hello"
#define LOW LOW  ", world"
```

The two terms to be expanded are: `glue(HIGH, LOW)` and `xglue(HIGH, LOW)`. The first simply results in `HIGHLOW` after substitution; and on the rescan, `HIGHLOW` is replaced by `"hello"`.

The second requires the expansion of the argument `LOW` to `LOW ", world"` before substitution. After substitution of `HIGH` for parameter `a` and `LOW ", world"` for parameter `b`, the result is: `HIGH ## LOW ", world"` and before the rescan, the `##` is deleted and tokens `HIGH` and `LOW` are concatenated giving: `HIGHLOW ", world"`. After rescan, this becomes: `"hello" ", world"`.

Summary

The preprocessing examples given in the draft standard illustrate some of the new facilities introduced by the draft. This note has given a fuller explanation of these examples for new readers of the draft.

Report on POSIX Meeting: 2-4 March 1988, London

Cornelia Boldyreff
Cornelia.Boldyreff@brunel.ac.uk

PRACTITIONER PROJECT
Department of Computer Science
Brunel University

Introduction

This was the inaugural meeting of WG15. It was attended by delegates from Canada, Denmark, the UK and the USA; as well as several observers and representatives from SC18/WG9 (User Interfaces) and SC21/WG5 (OSCRL). Jim Isaak, the IEEE P1003 Chairman and convenor of WG15, chaired the meeting; and Roger Martin of the NBS acted as secretary to the meeting. The meeting was held over three days.

Liaison Activities and Status Reports

The first day consisted of establishment of liaison activities with related standards work such as OSCRL, C language, Ada, and liaison activities with related organisations such as NBS, X/OPEN and */usr/group*. Brief reports were made of the status of related standards and the progress of associated IEEE activities. The table below summarises these:

Standard	Status
1003.1 - POSIX	Balloting
1003.2 - Shell and Utilities	1Q88
X3J11 'C'	2Q88
1003.3 - Test Methods	3Q88
1003.4 - Real Time	1Q90
1003.5 - Ada	2Q89
1003.6 - Security	1Q90
1003.0 - Guide to POSIX Based Systems	1Q89

With respect to the work of 1003.3, Roger Martin of the National Bureau of Standards spoke on their aim of defining what must be tested, i.e., test assertions and not how testing must be done. He mentioned the recent NBS POSIX FIPS indicating that this was an interim document, and that NBS intended to move to a FIPS based on the full use version of the POSIX standard when it was available. In conjunction with the FIPS, the NBS has developed the PCTS — POSIX Conformance Test Suite. The PCTS is based on the SVVS Subset from ATT. It will be available in source code form in the public domain; it will be distributed by the National Technical Information Service. NBS will maintain the test suite as the standard evolves. They aim to encourage use of the test suite by third party testing services.

The remainder of the first day was taken up with an introduction to work on User Interfaces by SC18/WG9 and SC21/WG5 OSCRL. Although the former work is not directly relevant to the current POSIX work; it was acknowledged that some elements of the OSCRL work placed constraints (e.g., security) on POSIX applications. The OSCRL work presents the user with a different system view and as such is complementary to POSIX. It was agreed that the WG would express a willingness to accept the OSCRL work item; this would allow the current OSCRL documents to be accepted as technical reports for consideration in the future work of WG15.

Main Items of Business on Days 2 and 3

The most pressing business of the meeting was to review the results of the SC22 Ballot and coordinate a response. The Ballot results consisted of only two votes of disapproval from Canada and Japan. The issues raised by Canada were resolved at the meeting; unfortunately, the Japanese comments were not available during the meeting. All other comments accompanying votes of approval from Denmark, France, Germany and the UK were processed. A Disposition of Comments document was drafted.

The WG discussed future work including a division of the work item (see Resolution 9 below); and the following dates of future biannual meetings were agreed:

20-21 Oct 88 -Tokyo
Apr 89 - Ottowa, Canada
Oct 89 - Brussels
Apr 90 - Paris
Oct 90 - USA

It was agreed that subject to satisfactory resolution of the Japanese comments, the WG should put forward DP9945 for registration as a DIS. It is anticipated that if DP9945 is registered as a DIS in June or July this year, the six-month ISO Ballot will have closed in good time before the WG meeting planned in the spring of 1989.

Resolutions of WG15

The text below is taken from my own notes; and may appear in a revised form in the official minutes.

Resolution 1

JTC1/SC22/WG15 requests that the US member body incorporate the recommendations made in and resolve the outstanding issues in JTC1/SC22/WG15/N___, "Disposition of Comments on DP9945" into the revised POSIX document.

Resolution 2

JTC1/SC22/WG15 requests that the US member body take into consideration recommendations made in JTC1/SC22/WG15/N___, "Disposition of Comments on DP9945", including the UK and French member bodies comments on language independent specifications when developing future versions.

Resolution 3

JTC1/SC22/WG15 recommends that the revised document adopted by the IEEE which incorporates responses to SC22 comments be registered as a DIS.

Resolution 4

JTC1/SC22/WG15 requests that the SC22 Secretariat coordinate determination of the proper document format and forward that information to the POSIX document editor as soon as possible.

Resolution 5

The following comments shall be submitted to the SC22 Secretariat regarding recommendations made in TC97/N1935:

1. The work item scope has been modified as proposed. JTC1/SC33/WG15 is committed to the development of a language independent specification of POSIX.
2. JTC1/SC22/WG15 met representations of the OSCRL committee and issues resolutions pertaining thereto.
3. The POSIX trademark has been dropped by IEEE.

Resolution 6

JTC1/SC22/WG15 requests that the US member body develop a single interchange format that resolves the concerns raised regarding CPIO and TAR, and provide such format for inclusion in a future version of the POSIX document. The existing definitions of both CPIO and TAR formats in POSIX should be retained until a new, single interchange format definition is available.

Resolution 7

JTC1/SC22/WG15 wishes to express its thanks to the SWG on POSIX, SSI and Related Issues. Following the recommendations of the SWG (TC97 N1935, Att. D.), the definition of the scope of work was clarified, and the POSIX work item was accepted and assigned to SC22 for development.

The WG also wishes to thank SC22 for its prompt action on the POSIX NWI in Sept 1987. By unanimously voting to accept registration of DP9945 allowing WG15 to proceed immediately to a DP Ballot, work on this standard has been greatly expedited.

Resolution 8 — Regarding OSCRL

JTC1/SC22/WG15 expresses to the the SC22 Secretariat its willingness to accept the OSCRL work item. If OSCRL is assigned to WG15, the current OSCRL documents would be accepted as technical reports for consideration in the future work of WG15.

Resolution 9

JTC1/SC22/WG15 instructs the WG15 Convenor to initiate a division of the work item, and submit this to the SC22 Secretariat for action. Said division should reflect the following WG15 activities:

1. Continued work beyond 9945 to address:
   - — Language Independent Bindings
   - — Ada and C Bindings
   - — Additional Service Definitions for Terminal Control, Data Interchange, etc
2. A separate standard document pertaining to the Operating System Environment based on the IEEE 1003.2 work shall be initiated.

3. Formation of a Rapatour Group on Security to do preliminary investigations of the scope and requirements for work in this area, and to coordinate with the IEEE 1003.6 effort.

WG15 will also need complementary work to be done in the areas of — testing, windows, application/human interfaces, systems administration, distributed systems, and network interfaces.

Resolution 10

JTC1/SC22/WG15 instructs the WG15 convenor to respond to the Japanese Comments on DP9945 in consultation with WG15 experts and prepare an addendum to the Disposition of Comments document.

More Obfuscated C Code

Mick Farmer
mick@cs.bbk.ac.uk

Here's a program we wrote recently. It originated in a discussion I had with Dave Tilbrook and my seeing the winners of the 'Obfuscated C Code Contest' for 1986.

```
**********************************************************************
#define 1000 char
#define 1001 mess
#define 1010 [
#define 1011 13
#define 1100 ]
#define 1101 =
#define 1110 {
#define 1111 'H'
#define 1001 ,
#define 1010 'e'
#define 1011 'l'
#define 1100 'o'
#define 1101 ' '
#define 1110 'w'
#define 1111 'r'
#define 1011 'd'
#define 1011 '0
#define 1101 ' '
#define 1101 }
#define 1110 ;
#define 1110 main
#define 1111 (
#define 1111 )
#define 1111 printf
1000 1001 1010 1011 1100 1101 1110 1111 1001 1010 1001
1011 1001 1011 1001 1100 1001 1101 1001 1110 1001 1100
1001 1111 1001 1011 1001 1011 1001 1011 1001 1101 1101
1110 1110 1111 1111 1110 1111 1111 1001 1111 1110 1101
```

UNIX Clinic

Colston Sanger
olibc1!colston@olgb1.oliv.co.uk

Olivetti International Education Centre

Colston Sanger is a lecturer at the Olivetti International Education Centre, Haslemere, UK and a visiting lecturer in the Faculty of Engineering, Science and Mathematics at Middlesex Polytechnic, London.
He is *not* the Musical Director of the Pangbourne & District Silver Band...

About this column

At the recent EUUG conference in London Alain Williams asked if I would like take over this column. I said ''OK'', so — here goes.

About this column: I see it as being primarily for the post–1984, UNIX System V generation. That's to say, I see it as being for users who want or need to use the UNIX System to accomplish practical, real–world tasks. Most of you (or us — because I'm also one of the post–1984 generation) are working in the commercial rather than academic world, and have UNIX binary as opposed to source licences. In short, then, it's for beginners rather than people who use `cat` to write device–drivers.

As to the flavour of UNIX, the emphasis is likely to be on System V: not only because that's ''The Right Choice'' (as the advertisement says), but also because that's the kit available to me. Where I work — at the Olivetti International Education Centre — we have a Starlan network of Olivetti–AT&T 3B2's running UNIX System V Release 3.1 and RFS, together with various Olivetti PC's running MS–DOS and the AT&T DOS Server package, but that's all. Sorry, but I just don't have a VAX.

That said — the introductory stuff, I mean — let's get into it.

Screen management in shell

There's been a lot of talk lately about user interfaces: News, X, Andrew, Open Look and so on. I thought it might be interesting to look at what can be done using plain old `/bin/sh`.

The basics — `echo` and `read`

First a quick review of the basics: `echo` and `read`. You know what `echo` does, but the System V shell built–in `echo` also takes a set of special escape characters:

`\b`	backspace
`\c`	do not append a newline
`\f`	formfeed
`\r`	carriage return
`\t`	tab
`\v`	vertical tab
`\\`	backslash
`\0n`	where `n` is the eight–bit character whose ASCII code is the one–, two– or three–digit octal number representing that character.

For example:

```
$ echo "\n\tPlease enter name: \c"

          Please enter name: _
```

(The underscore here is meant to be the cursor on your terminal screen.) Berkeley people will have to use

```
$ echo -n "\n\tPlease enter name: "
```

instead of the quoted `\c`.

`read` is pretty simple too. Within a shell script, just:

```
echo "\n\tPlease enter name: \c"
read NAME
```

where `NAME` is a shell variable.

Using screen attributes — `tput`

Suppose you want to use screen attributes such as reverse video, blink, underlining and so on? This is where it gets interesting — and also where the System V `tput` command comes in.

Try:

```
$ tput blink
$ echo Hello world
```

OK, if you can't stand it any longer, type:

```
$ tput sgr0
```

There's a complete list of terminal attributes (and potential arguments to `tput` — depending on the capabilities of your terminal) under *terminfo*(4) in the UNIX System V Programmer's Reference Manual. For now, some useful attributes are:

`clear`	clear the screen and leave the cursor at the top–left (home) position
`bold`	turn on bold (extra–bright) mode
`smso`	begin standout (reverse–video) mode
`rmso`	end standout mode
`smul`	begin underscore mode
`rmul`	end underscore mode
`blink`	turn on blinking
`dim`	turn on dim (half–bright) mode
`bel`	ring the terminal bell
`sgr0`	turn off all attributes.

Note for Berkeley people: I don't think you have an equivalent to the System V `tput` command, although there are public–domain versions of `tput` around that use the `termcap` database.

A note of style

It's good style to assign screen attributes to shell variables (by command substitution) at the beginning of your shell script as in this example — a companion for the classic `phone`, everybody's first shell script:

```
# phone.add - enter phone nos in $HOME/lib/phone.nos

CLEAR=`/usr/bin/tput clear`
STANDOUT_ON=`/usr/bin/tput smso`
STANDOUT_OFF=`/usr/bin/tput rmso`

YN=YES
while [ "${YN}" = "YES" ]
do
        echo "${CLEAR}Ohone.add v1.0\n"
        NAME=
        echo "\t${STANDOUT_ON} Name: ${STANDOUT_OFF} \c"
        read NAME
        echo "\n\t${STANDOUT_ON} Address: ${STANDOUT_OFF} \c"
        ADDRESS=
        while read A
        do
                echo "\t\t     \c"
                case ${A} in
                        '')     # blank line ends address
                                break ;;
                        *)      ADDRESS="${ADDRESS} ${A}" ;;
                esac
        done
        echo "\n\t${STANDOUT_ON} Phone: ${STANDOUT_OFF} \c"
        read PHONE
        echo "${NAME}\t${ADDRESS}\t${PHONE}" >> ${HOME}/lib/phone.nos
        echo "\n\tAdd another name (Y/N) ? \c"
        read YN
        case ${YN} in
                [Yy]*)  CONTINUE=YES ;;
                *)      CONTINUE=NO ;;
        esac
done

sort -u ${HOME}/lib/phone.nos -o ${HOME}/lib/phone.nos
```

Why is it good style? Well, first, it parameterises things: if you want to use underline instead of reverse–video for highlighting you only need to change two lines at the top. Second, it's more efficient than

```
        echo "\t`tput smso` Name: `tput rmso`\c"
```

because you're not spawning an extra process for every `tput` command: just echoing a shell variable.

Making menus

Given `echo`, `read` and `tput`, it's easy to make menus — though it has to be said that genuinely 'user–friendly' menus are quite hard to do. Here is the outline of a fairly traditional menu:

```
# monitor - main menu
# trap interrupts
trap ' ' 1 2 3

# set standard variables
MACHINE=`/bin/uname`    # machine name
BIN=${HOME}/monitor/bin   # dir where progs are
```

```
export MACHINE BIN

# set screen attributes
CLEAR=`/usr/bin/tput clear`
STANDOUT_ON=`/usr/bin/tput smso`
STANDOUT_OFF=`/usr/bin/tput rmso`
NORMAL=`/usr/bin/tput sgr0`
BELL=`/usr/bin/tput bel`
export CLEAR STANDOUT_ON STANDOUT_OFF NORMAL BELL

HEADER="${CLEAR}${STANDOUT_ON} ${MACHINE} monitor
                `/bin/date` ${STANDOUT_OFF}"
export HEADER

while :
do  echo "${HEADER}\n\n\n"
    echo "\t\t\t1 - Usage monitoring"
    echo "\t\t\t    (whodo, who -u, ps -ef.)\n"
    echo "\t\t\t2 - Performance monitoring"
    echo "\t\t\t    (Run sar, print or "
    echo "\t\t\t    display results.)0
    echo "\t\t\t3 - System accounting"
    echo "\t\t\t    (Print daily/monthly"
    echo "\t\t\t    reports.)\n"
    echo "\t\t\tq - Quit.\n\n"
    echo "\tPlease type the number of your selection"
    echo "\tand press the RETURN key  > \c"
    read SELECTION
    case ${SELECTION} in
        1)  ${BIN}/monitor.1 ;;
        2)  ${BIN}/monitor.2 ;;
        3)  ${BIN}/monitor.3 ;;
        [Qq]*)  exit ;;
        !*) COMMAND=`echo ${SELECTION} | cut -c2-`
            eval ${COMMAND}
        echo "\n\tPress return to continue > \c"
        read RESPONSE ;;
        *) echo "\n\t${BELL}Unknown option - please try again."
           echo "\n\tPress return to continue > \c"
           read RESPONSE ;;
    esac
done
exit 0
```

Form–filling

Another style of interface that is sometimes called for is form–filling. Here, for example, is an outline for a form–filling front–end to pr, the draft formatter:

```
# fpr - form-filling front-end to pr

CLEAR=`/usr/bin/tput clear`
REVERSE=`/usr/bin/tput smso`
UL=`/usr/bin/tput smul`
NORMAL=`/usr/bin/tput sgr0`
```

```
LINE="------------------------------------\
------------------------------------------"
COMMAND=/bin/pr

trap 'echo $CLEAR ; exit' 0 1 2 3

echo "${CLEAR}\n${REVERSE} pr - draft \
formatter ${NORMAL}\n\n"

echo "${LINE}"

while test -z "${FILE}"
do
      echo "\n\t${UL}Please enter \
filename${NORMAL}................ \c"
      read FILE
      case ${FILE} in
            '')     /usr/bin/tput bel
                    /usr/bin/tput cuu1
                    /usr/bin/tput cuu1 ;;
             *)     break ;;
      esac
done

echo "\n\t${UL}Title to use (RETURN for \
filename)${NORMAL}... \c"
read TITLE
case ${TITLE} in
      '')    ;; # default is filename+date+page-no
       *)    COMMAND="${COMMAND} -h ${TITLE}" ;;
esac

echo "\n\t${UL}Page length (66)${NORMAL}......\
............... \c"
read P_LNGTH
case ${P_LNGTH} in
      '')    ;; # default is 66 lines per page
       *)    COMMAND="${COMMAND} -l${P_LNGTH}" ;;
esac

echo "\n\t${UL}Line length (72)${NORMAL}......\
............... \c"
read L_LNGTH
case ${L_LNGTH} in
      '')    ;; # default is 72
       *)    COMMAND="${COMMAND} -w${L_LNGTH}" ;;
esac

echo "\n\t${UL}Single or double-spaced \
(S/D)${NORMAL}........ \c"
read SPACING
case ${SPACING} in
      [Dd]*)     COMMAND="${COMMAND} -d" ;;
          *)     ;;
```

```
esac

echo "\n\t${UL}Send output to \
(Terminal)${NORMAL}........... \c"
read PRINTER
case ${PRINTER} in
          # default is the terminal
    '') ;;
          # pipe the output to PRINTER
     *) PRINTER=" | lp -d${PRINTER}" ;;
esac

echo "\n${LINE}"

echo "\n${REVERSE} Command is:\
 ${NORMAL}${COMMAND} ${FILE} ${PRINTER}"
echo "\n\t${UL}Execute (Y/N)${NORMAL}? \c"
read X
case ${X} in
    [Yy]*)     eval "${COMMAND} ${FILE} ${PRINTER}" ;;
        *)     ;;
esac
exit 0
```

This is no more than an outline. Obviously, it would be helpful to know early on that the file to be formatted exists and is readable.

Cursor addressing

In UNIX System V Release 3 you can type:

```
$ tput cup 10 5
```

to move the cursor to row 10, column 5. In UNIX System V Release 2 it's not quite so easy.

There are two ways of doing cursor addressing in UNIX System V Release 2. The first way is fast, but not really to be recommended because you have to hardcode the terminfo `cup` (cursor addressing) capability within your shell script. The second way is to call a C program.

The first way (in System V Release 2)

Here is an example of the first way, the hard way:

```
# cursor_pos - cursor addressing the hard way

# parameterised cup capability
case $TERM in
        vt100)  begin=^[[; mid=';'; end=H ;;
        50)     begin=^[a; mid=R; end=C ;;
esac
tput clear
echo "\n\tPlease enter ROW: \c"
read ROW
echo "\tPlease enter COL: \c"
read COL
echo "${begin}${ROW}${mid}${COL}${end}Hi there! \c"
exit 0
```

What this means is that to move the cursor to a selected row and column on a vt100 (for instance), you have to send the terminal:

```
  ESC [

  followed by the row number
  (in decimal)

  followed by a ;

  followed by the column number
  (in decimal)

  followed by an H
```

How do you know this? Because you have looked it up in the terminal manual, or because you have tried typing

```
  $ tput -Tvt100 cup | cat -v
  ^[[%i%p1%d;%p2%dH
```

And to find out what *that* means, you'll have look up *terminfo*(4) in the UNIX System V Programmer's Reference Manual.

The second way (in System V Release 2)

The second way is to call a C program. For example:

```
# del_reg - delegate registration
#           cursor addressing example

PS1=
CURSOR='/usr/bin/tput cup'
CLEAR='/usr/bin/tput clear'
UL='/usr/bin/tput smul'
REVERSE='/usr/bin/tput smso'
NORMAL='/usr/bin/tput sgr0'
BEL='/usr/bin/tput bel'
export CURSOR

# display screen form
echo "${CLEAR}"

cursor 1 25 ; echo "${UL} Delegate Registration\
 ${NORMAL}"
cursor 1 65 ; echo "'/bin/date '+%d %m·%y''"
cursor 4 5 ; echo "First Name: ..............."
cursor 4 35 ; echo "Last Name: ........................"
cursor 6 10 ; echo "Home address: .....................\
............"
cursor 7 24 ; echo "................................."
cursor 8 24 ; echo "................................."
cursor 10 10 ; echo "Post code: ........"
cursor 10 46 ; echo "Tel. no: ............."
cursor 13 10 ; echo "Car reg. no: ........."
cursor 16 5 ; echo "Company: ........................\
..................."
cursor 19 5 ; echo "Course: ............................"
cursor 19 46 ; echo "Room no: ...."

# read input
```

`cursor` is the C program.

Speeding things up

If you try `del_reg` as it is, it will run quite slowly. However, one simple trick will make it much faster:

```
   $ del_reg > dr2.scr_vt100
```

So now, here is version 2 of the *del_reg* script:

```
# del_reg2 - delegate registration (version 2)
#            cursor addressing example

PS1=
CURSOR='/usr/bin/tput cup'
BELL='/usr/bin/tput bel'
export CURSOR

LIB=${HOME}/lib

# trap interrupts
trap ' ' 1 2 3

# display screen form
if [ "${TERM}" = "vt100" ]
then
      cat ${LIB}/ dr2.scr_vt100
elif [ "${TERM}" = "50" ]
then
      cat ${LIB}/ dr2.scr_50
else
      echo "Sorry, this example only works with vt100"
      echo " or Wyse 50 terminals" 1>&2 ; exit 1
fi

# read input
cursor 4 17 ; echo "${BELL}\c" ; read F_NAME
```

Finally, here is `cursor.c`. I stole it, then modified it slightly from Rod Manis and Mark H.Meyer's *The UNIX Shell Programming Language*, Howard W.Sams, 1986 (ISBN: 0–672–22497–6), which I think is the best of the four books about the shell (`sh`, `csh` and `ksh`) published in the last eighteen months. The good news for Berkeley people is that it should also run on your system.

```
/*
* cursor.c - move the cursor to a specified
*            line and column on the screen.
*
* cursor is intended to be used within a
* shell script to provide cursor addressing.
* So it's very handy for constructing
* prototype data-entry screens.
*
* Normal usage would be (within a shell script):
*
```

```
*     PS1=
*     CURSOR=`tput cup`
*     ...
*     cursor 5 10 ; echo "Please enter your name: \c"
*
* Alternatively (from the command line):
*
*     cursor 5 10 `tput cup`
*
*
* Compile as: cc -O -s -o $HOME/bin/cursor cursor.c -lcurses
*
* or (for Berkeley systems)
*             cc -O -s -o $HOME/bin/cursor cursor.c -ltermlib
*/

#include <curses.h>
#include <term.h>

#define USAGE  "Usage: cursor line col0
#define EUSAGE 1
#define NOCURSOR "%s: no CURSOR argument or \
                            $CURSOR shell variable.0
#define ENOCURSOR 2
#define LINE 1
#define COL 2
#define CURSOR 3

char *tgoto();
char *getenv();
main(argc, argv)
int argc;
char *argv[];
{

    char *cup;

    setupterm(0,1,0);

    if (argc < 3) {
        fprintf(stderr, USAGE);
        exit(EUSAGE);
    }
    else if (argc == 3) {
         if (cup = getenv("CURSOR")) {
        /*
         * tgoto is an old function, included for
         * compatibility with termcap.
         */
         printf("%s", tgoto(cup, atoi(argv[COL]),
                             atoi(argv[LINE])));
         } else {
            fprintf(stderr, NOCURSOR, argv[0]);
            exit(ENOCURSOR);
            }
        } else {
        printf("%s", tgoto(argv[CURSOR], atoi(argv[COL]),
                                      atoi(argv[LINE])));
    }

    reset_shell_mode();
    resetterm();
    exit(0);
}
```

In the next issue

That's it for this column. Let me know (by e-mail) whether you find this stuff interesting or useful. In the next issue, unless anyone suggests a better topic, I'm planning to write about efficiency considerations and shell functions in the System V shell.

Found on the Floor ...

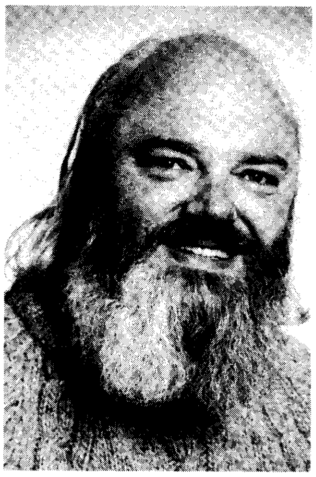
Found by Mick Farmer on the floor of the QEII conference centre ...

```
 ________________________________________________
/WELL DONE SUNIL. AN EXCELLENT CONFERENCE        |
| ▯   ▯  ▯    ▯ ▯ ▯  ▯ ▯▯  ▯   ▯  ▯▯ ▯ ▯▯        |
|  ▯▯  ▯▯    ▯ ▯   ▯     ▯▯ ▯   ▯▯  ▯ ▯          |
|▯         ▯▯         ▯      ▯                   |
|11111111111111111▯111111111111111111111111111111|
|2222222222▯2222222222222222222222222222222222222|
|33▯▯3333333333▯▯333333▯3▯▯33▯3▯3333333▯333333333|
|44444▯44444▯444444444444444444444444444444444444|
|5▯55555▯▯555▯55555▯5▯55▯55▯▯5555▯5▯5▯▯5▯55555555|
|▯66666▯666666666666666666666666▯6▯66666666666666|
|777777777777777777777▯77777777777777777777777777|
|888888888888888▯8888888888888888888▯888888888888|
|9999999999999▯9999999999999999999999999999999999|
|________________________________________________|
```

C++ In Print

John Carolan
john@puschi.uucp

Glockenspiel Ltd.
Dublin



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.

In March '88, the only book you could buy on C++ was Bjarne Stroustrup's.

Come April, you're not exactly spoilt for choice, but clearly the deluge is beginning.

First of all, there's conference proceedings.

The Santa Fe conference on C++, I reviewed last issue. Also included in the conference proceedings are a few really good papers that the editor thought would interest people...

Experience in Using C++ for Software System Development

by Bill Hopkins

An excellent paper by a real end-user of C++. Bill's team at AT&T in Denver built a very large software product in C++ before anyone had heard of C++. The paper describes C++ concisely and enumerates the benefits that accrued from using it. We badly need more papers like this from practitioners of object-oriented programming.

Possible Directions for C++

by Bjarne Stroustrup

Catalogues the opportunities which the Perpetrator sees for improving the language. Top of the list are 'parameterised types' (which would give to C++ classes a functionality similar to generics in Ada) and 'exception handling' (which would improve the interaction of `setjmp/longjmp` constructs with scoped operations such as destructor invocation).

A Set of C++ Classes for Co-Routine Style Programming

by Stroustrup & Shopiro

Describes a working implementation from the Labs. Rumor: release 2.0 of AT&T's C++ will have a better 'tasks' library.

C++ vs. Lisp

by Howard Trickey

Compares by code-size, compile-time and various similar metrics the same slightly expert graphics software done in C++ and LISP. In my opinion, it is totally worthless because:

1. The LISP used was an obscure deviant from Common LISP;

2. The architecture of the program was preserved instead of being redesigned to use inheritance and polymorphism Despite the unreasonable basis for comparison, the paper comes out strongly in favor of C++.

Several other papers included are of the general form "Let's hack on the C++ language to do X". For the present, I believe the only guys who should hack on the language are Bell Labs — otherwise we will end up with a disaster instead of a standard language.

Next, on the subject of conferences, the EUUG conference in London included many papers describing projects implemented in C++. The proceedings include three papers ostensibly about C++.

Yacc Meets C++

by Steve Johnson

This paper presents an extension to yacc to allow inheritance in `yacc` grammars.

Formatted I/O in C++

by Mark Rafter

This paper illustrates magnificently how you can add functionality to C++ by elegant extension of an existing class library — in this case 'streams'.

Software Re-Engineering using C++

by Anderson & Gossain

Compares the same program in C and C++. The approach here was to redesign a graphics presentation program in an object-oriented way and implement it in C++. The comparison consists of both qualitative issues such as re-usability and metrics such as a count of magic numbers present in the code.

Now, on to the first text-book on C++ since 1986.

An Introduction to Object-Oriented Programming and C++

by Wiener & Pinson
Addison-Wesley
ISBN 0-201-15413-7

The book presents a good introduction to the jargon of OOP, at least until it gets to page 8. On page 8 it presents the first of many serious mis-understandings. The book tries to describe the process of designing an inheritance hierarchy. It uses for an example an automobile with components such as engine and gear-box. It implies that engine and spark-plug should be derived from automobile, which misses the following point: Functional decomposition of a problem and class derivation are orthogonal design processes. Both processes must be pursued to achieve a good solution. The book also confuses bottom-up design with specialisation. Apart from the view of OO design being seriously damaged, the book does not consider that OOP should extend to human interface issues at all. Most of the worked examples come from older books on algorithmic programming. Hoare's QuickSort algorithm and much list-processing code occur in the book. Why I object to these examples is that they dissipate the reader's energy on following the algorithms. It must be possible to choose examples which illustrate the OO approach to design without so much implementation details. Examples which deal with human interface to some extent would be more interesting and could show how OOP assists consistency in the human interface.

As a textbook on C++, W&P has serious flaws. I counted over twenty errors. Here's an example from page 126:

> "The public qualifier in the derived class definition indicates that objects of the derived class can use all the methods of the parent class, unless these methods have been redefined in the derived class."

Stroustrup's book may be hard going, but at least it is accurate. Nowhere does W&P offer a Rosetta Stone for translating between C++ terminology and OOP terminology. It calls all data members of classes 'fields', for example. Now, in C and C++, only members with the form:
`< declarator : constant-expression >`
are called fields. It confuses objects with classes and declarations with definitions in several places. For example, on page 21:

> "The operators in the cin, cout and cerr classes can be overloaded within a programmer-defined class."

W&P is very misleading on the interaction between `typedef`, in built types and classes. The book omits several important features of C++. The most noticeable being programmed assignment, programmed conversion, coercions and pointers to member functions.

The last chapter contains three example programs. The first is a spelling checker. In it, there is a class 'word' which has a member function '`open_file()`'. Now, how can a word have a behaviour such as open file? In a book which attempts to instruct people on OOP, this is unacceptable. There should be a class such as WordFile on which Word is not dependent. The second example is a golden oldie from Simula which does an event-driven simulation of queues in a bank. The book follows the OOP approach nicely until it comes to the human interface. Then ZAP!, objects are nowhere to be seen. The output from the program is very difficult to interpret simply because the OOP paradigm gets dropped. The third example is a simple function Interpreter. It parses a calculator language using an expression tree and it does syntax checking using a finite state machine. The syntax checker does not re-use any of the code in the parser. However, the program is riddled with friend declarations, which weaken the abstractions unnecessarily. The finite state machine declares each state as a separate class with one instance, which immediately rings warning bells. Well, the warning bells are correct. It is simpler to code this particular FSM in a table-driven form. The classes should be state transition tables, not states. If you are in any doubt about my criticism of this example, then repeat the following exercise: Add a unary minus operator to the Function Interpreter and count in how many places you have to change the code. Lastly, this example restricts the user interface just because W&P want to use operator overloading in a place where it is not appropriate. The user of the program types a formula where the arguments are mentioned explicitly. Each invocation of the formula asks the user to say how many arguments again. Not a good user interface.

In summary, the book covers neither nor C++ very well. It needs a lot more proof reading by practitioners of C++ before I would recommend it to someone learning C++.

Tech Tip

Here is a posting from Jonathan Shopiro which describes the usage of assignment and initialisation operators very neatly...

If a copy-initialiser is not supplied by the programmer, the compiler generates the default version, which looks like the following for a class with arbitrary base classes and members:

```
class D : B1, B2 ... {
        T1      m1;     // m1 is a data member, T1 is a type
        T2      m2;
        ...
};

D::D(const D& d) : B1(d.B1), B2(d.B2) ... m1(d.m1), m2(d.m2) ... {}
```

Each initialisation is the copy-initialisation for the appropriate type. They are executed left to right. Of course, such copy-initialisation is usually optimised to ''blast the bits from the source to the target''. If you leave the `const` specification out, you won't be able to copy a constant object. Following the same principles, the default assignment function is:

```
D&
D::operator=(const D& d)
{
        *(B1 *)this = (B1 &)d;
        *(B2 *)this = (B2 &)d;
        ...
        m1 = d.m1;
        m2 = d.m2;
        ...
        return *this;
}
```

It too is usually optimised to ''blast the bits.''

STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS ST
OP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS

This year's C++ conference will be in Denver, Colorado October 17 to 20. If you would like to submit a paper then send a 2 to 4 page abstract to:

Andy Koenig
fax: +1 201 580 4127
uucp: ....!research!ark

before June 14th

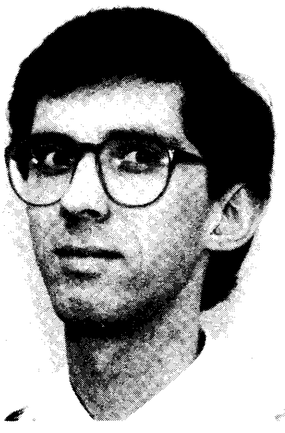
STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS ST
OP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS

Price for proceedings

The Santa Fe conference proceedings are available from the EUUG secretariat (Owles Hall) at a price of £30 includings postage & taxes.

This is the best read around on C++ that you will find in Europe. There are only a limited number of copies — don't be disappointed.

EUUG Tape Distributions



Frank Kuiper
mcvax!euug-tapes
euug-tapes@cwi.nl

Centre for Mathematics and
Computer Science,
Amsterdam

```
  _] [___|  |
 <__________|-1
    o-o-o
```

Once again the announcement of a new tape. This tape, the 'London' or EUUGD13 tape, so meticulously composed by Bjorn Ericson, did not reach London in time to be copied for the conference. Stuart McRobert did a fine job in assembling a tape which looked just as it should have been, and managed to make 130 copies. We had more requests for a conference tape than ever before.
A problem with the tapes seems to be their alleged non-readability. It has been suggested that the electro-magnetic security gates in the QEII-conference centre were the blame for this. So far I have had some five tapes returned, but if any of you are also having problems reading the conference tape, please return it to me (at the above address), and I will supply you with a new copy.

Due to the fact that the original tape did not reach London in time, there now are more or less two London tapes. So as not to make life to complicated, I will use the original tape as being the official London tape. The differences in contents are not that big. Anyone who received the tape issued at the conference, and who insists to obtain the official tape, can send me their conference tape, and I will supply them with a copy of the official tape. This is a one-time-only offer, and due to handling and shipping, I'm afraid I have to charge DFl 50,- for this service.

This brings me to another small problem that I encountered. Some people who order tapes from me, supply a cheque with their order for what they think is the right amount of money. Though the amount on the cheque often is near to the real price of the ordered goods, most of the time it isn't. Therefore I would strongly ask you not to send me any cheques, bank-orders or what ever. The moment that I send the tapes off to you, I inform the financial department of my institute, and they will, in time, send you an invoice for the goods. Especially since now that I also provide distributions on 1/4" cartridge streamer tape, prices for tapes differ widely. This is due to the higher price of cartridges.

One last remark. Due to the nature of my regular job, I am not always able to handle requests for tapes immediately. I admit that the introduction of the cartridge streamer tape as a new medium for the EUUG software distributions has slowed me down in handling orders. I regret any inconveniences that the late delivery of tapes may have caused or will cause.

The rest of this article consists of the well known list of distributions that are now available on both 1600 bpi reel-tapes, and QIC-24 format streamer-tapes.
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 (April 1988) 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 the EUUG Newsletter.
Prices of the tapes are in Dutch guilders (DFl), and do not include VAT-taxes or postage. The first price listed is for reel-tapes, the second one is for distributions on cartridge tapes.
Note 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,-/180,-
EUUGD2:	Early Pascal compiler of the Free University of Amsterdam. V7 source licence minimum. Price: Dfl 120,-/180,-
EUUGD3 R3:	Currently under not available.
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: Dfl 150,-/130,-
EUUGD5:	A collection of benchmark programs made up by EUUG. Price: Dfl 60,-/120,-
EUUGD6:	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: Dfl 240,-/300,-
EUUGD7:	UNIX|STAT Version 5.2. A collection of about 25 data manipulation and analysis programs written in C by Gery Perlman. Price: Dfl 60,-/120,-
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,-/210,-
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,-/150,-
EUUGD11:	This is the 'Boat' tape; the Helsinki EUUG 1987 spring conference. It contains about 25 Megabytes of programs, games, etc. Including: jove, less, nag, news, rn, µEmacs, uuencode and larn. Price: Dfl 120,-/180,-
EUUGD12:	This is the Dublin EUUG 1987 autumn conference tape. It contains about 26 Megabytes of programs, games, etc. Including: copytape, crc_plot, fastgrep, jove, kermit, notes, uupc, nethack, cron, sendmail, mh, Recipes, brl-gw, isode, pcip, pctelnet. Price : Dfl 120,-/180,-
EUUGD13:	The latest conference tape for the London EUUG 1988 spring conference tape. A table of contents follows:

EUUG.D13 TABLE OF CONTENTS

3. C Library Subroutines

`pc-curses`	screen/window management library
`regexp`	regular expression handler
`rpc`	library routines for remote procedure calls
`syslog`	logging routines

4. Special Files

`pty`	pseudo tty driver for system V machines

5. File Formats

6. Games

`moria`	4.85, a dungeon adventure game in the manner of rogue
`omega`	another adventure game

7. Miscellaneous

8. System Maintenance

`arc`	a general archive utility
`autobaud`	terminal speed detection
`backup`	perform tape backups
`enable`	enable, disable getty on tty lines
`smail`	UUCP mailer with routing
`survey`	generate simple plot of system load and # of users
`sush`	restricted shell to grant specific limited privileges
`watcher`	system monitoring program

Price : Dfl 120,-/180,-

Ordering EUUG Tapes

If you want to order a 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.uucp (or: frankk@mcvax.uucp)

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 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 an extra Hfl 300,- per tape as a handling fee.
Please enclose a copy of your membership or contribution payment form when ordering.
Do not send any money or cheques, you will be invoiced. Tapes and bill will be sent separately.

All reel-tapes come in tar format, 1600 bpi. 800 bpi is possible on request. Cartridge tapes come as tar image, written with dd, with a blocking of 126b. This is a so-called QIC-24 format. QIC-11 is available on request.

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 declare to 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."

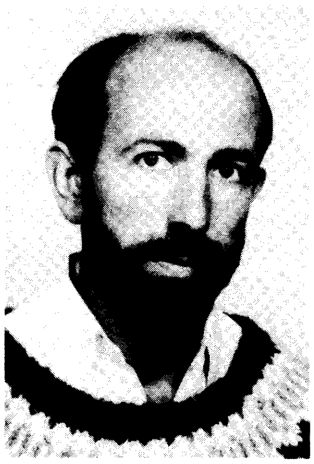
Signature: ....

Date: ....

EUnet in Finland

Peter Houlder
uknet@ukc.ac.uk

Computing Laboratory,
University of Kent



Introduction

This time we are grateful to Finland for this article. Juha Heinanen has provided a clear interesting account of the Finnish network. We now have had details on Finland, France, The Netherlands, Sweden, The United Kingdom and West Germany along with details of the overall network structure. Hopefully the other missing EUnet countries will soon fill in gaps, but other articles on:

Type of Mailers
Mail Interfaces
File Transfer
X400
News Software
Local and Wide Area Networks

and any other aspect of networking would be very welcome. Please send all contributions to the above (uknet@ukc).

Juha Heinanen
jh@tut.fi

Tampere University of Technology
Finland

Juha Heinanen is currently an associate professor in computer science at Tampere University of Technology in Finland. Networking has been his hobby since 1983 when we established the EUNET link from University of Tampere to Enea Data Ab in Sweden. Juha is also member of the steering committee of the Finnish University and Research Network Project (Funet).

EUnet in Finland

History

This all began in 1983 when a UUCP mail link was established from University of Tampere to Enea Data Ab in Sweden. The first UUCP machine was a PDP 11/34 and, of course, its address/disk space was too small to run the news software.

In 1984 Helsinki University of Technology (HUT) got a link to mcvax and the enea link was transferred from Tampere University to the neighbouring

Tampere University of Technology (TUT). The news was received by TUT from enea while HUT handled mail with mcvax. In 1985 FUUG (Finnish UNIX Users' Group) established Finland's first official EUnet backbone at Penetron Oy, which in 1986 also started to receive the news directly from mcvax. Mainly because of shortage of 'free' manpower at Penetron, the official backbone was moved in the beginning of 1987 to TUT.

TUT now handles international connections and serves the 'rural' parts of the country. The sites in the Helsinki area are served by HUT that has taken the responsibility of a second backbone.

Rate of Growth

During the five years of existence, EUnet in Finland (SFNET) now consists of 46 sites 11 of which receive the news. The main reason for the low number of news sites is the high cost, which currently is around $200/month.

The number of sites has grown steadily at the rate of some 10 sites/year. The new sites are almost exclusively commercial companies, typically software houses, since almost all of the universities are already subscribing to SFNET.

A new development at universities has been the installation of Internet domain naming in computer centre mainframes running VMS and VM operating systems. This has increased the international through traffic at TUT considerably, since earlier SFNET services were mainly used by computer science departments.

At the time of writing, 25 organisations have registered an Internet subdomain under .fi.

Backbone Personnel

At TUT, the mail and news service is now in practice run by Vesa Keinanen and I'm proud how fast he has become a real guru, mastering the mysteries of sendmail and other networking software. As a backup, we have Hannu-Matti Jarvinen who has several years' SFNET experience. At HUT, the work is done by Jukka Virtanen who runs a machine called santra. Jukka also maintains the EARN and CSNet gateways. Currently SFNET is totally uncommercial, i.e., we don't charge the sites for any of the salaries or other overhead. If such costs were added on top of already high news costs, the network would probably become too expensive for some of our current subscribers.

Links with Other Networks

SFNET has direct gateways to FUNET's (Finnish University and Research Network) TCP/IP and DECNet networks, EARN, CSNet, and the RARE experimental X.400 network. All other networks are reached via mcvax or enea.

Future Plans and Problems

All the universities in Finland will soon by linked by 64Kb leased lines managed by FUNET. For SFNET this means division of the sites into two quite different groups: the commercial companies running UUCP over X.25 or telephone lines, and the academia running LAN-protocols over the fast backbone.

So far all SFNET traffic has been allowed free of charge on the FUNET backbone. In the future, some kind of formal agreement might be needed between SFNET and FUNET concerning the through traffic. The relations between SFNET and FUNET are very good and there has been no pressure from either side to take over other's operations or compete with them.

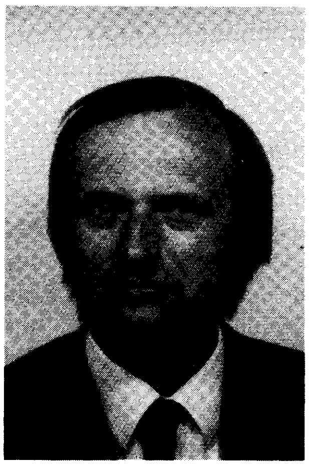
For the future benefit of SFNET, it might be desirable to find a wealthy commercial company that would take responsibility to develop further the non-academic part of SFNET. An ideal site would be one that has a large research department with a lot of contacts with both universities and software companies.

At the time of writing there are very interesting developments going on internationally, which hopefully will lead to a better and cheaper EUNET infrastructure within Scandinavian countries and also to Central Europe. The next report will tell how this all worked out.

X/Open Midterm Report

John Totman
Director of European Program

X/Open Company Ltd.
Lovelace Rd
Southern Industrial Estate
Bracknell
Berks RG12 4SN
UK
+44 344 424842 Extn 2748



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 and became a convert to the cause of standards and applications portability.

John is now a full time employee of the X/Open company and is responsible for European marketing activities.

Opening wide the X/Open Door

As we planned, X/Open has now widened its industry influence and involvement in two ways: by increased share holder memberships and by the establishment, in 1987, of our User and ISV Advisory Councils.

In terms of shareholders, NCR and Sun Microsystems joined in January 1988 bringing out total membership to 13 of the world's largest computer vendors. Our continuing growth in membership is acknowledgement of the unstoppable market trend towards vendor independent computing.

Our advisory councils, formed from major users such as General Motors, Eastman Kodak, British Airways, Daimler Benz, and software vendors such as Uniplex, Informix, Oracle, Quatratron, and Multihouse give direct advice to X/Open on the future development and expansion of the Common Applications Environment. Both these Councils meet regularly with X/Open and have already provided valuable input to our future plans and strategies.

In addition to the ISV Council, which can only be a small group to be fast moving and responsive, we have also established a programme of wider software vendor participations in X/Open by means of our Software Partnership scheme. This scheme is designed to give direct support to software vendors though out Europe and the USA who wish to produce and market software to the X/Open standard.

Software vendors wishing to take part in the scheme should contact me at the above address.

Technical "Hot Spots"

On the technical front, POSIX has been a major area of activity for the X/Open Technical Committees. By the time that this is published, we expect POSIX (i.e., P1003.1) to be formally endorsed as the official full use IEEE standard. On this basis X/Open will be incorporating POSIX functionality into the 1988 edition of the *Portability Guide*.

In addition to POSIX convergence X/Open has also developed a standard transport interface definition for networking (XTI) which provides a protocol independent system program transport interface; a user interface (a windowing standard based on X-

Window); the COBOL definition has been updated from a COBOL 74 to a COBOL 85 base; and the the GSA standard ADA has been adopted. All these will be published in the 1988 edition of the Portability Guide.

1988 will also be the year that verified X/Open conforming systems will be available in the market. Although compliant systems from X/Open members have been available since early last year it is only now that formally verified and branded systems will be available. This has been made possible by the formulation of conformance guidelines, establishment of public verification centres, and, later in the year, licencing of the X/Open system supplier to test for compliance to the X/Open standards and to brand compliant products.

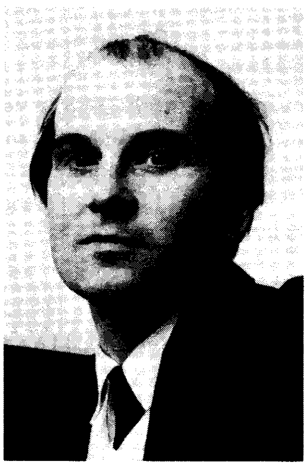
1988 Outlook

An aggressive marketing activity in the USA, coupled with the establishment of the X/Open Software Partners programme, is now attracting a wide base of industry support for X/Open standards. It is out expectations that by the end of 1988 major users, both in the US and Europe, will be specifying the X/Open Common Applications Environment as a procurement standard, confident that compliant products will be available from many suppliers.

Open Comments

X/Open now publishes a quarterly magazine. This is called *Open Comments* and is for those who wish to know more about developments within X/Open and its standards. If you would like to receive a copy please contact me.

Receiving News at a Small Commercial Site: Is It Worth It?



Dominic Dunlop
domo@sphinx.co.uk

Sphinx Ltd.
Maidenhead
United Kingdom

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.

My company, Sphinx Ltd., subscribes to Usenet, the UNIX world's distributed bulletin board system[1]. In the past, the net's user base has been dominated mainly by academic institutions, and by large, technically-oriented commercial companies. Today, however, more and more small companies are subscribing, even though the subscription involves significant costs, both in hard cash and in staff commitment.

Why is this? Trying to answer this question — in response to a query which had appeared on the net — I came up with the following summary of the benefits, the costs — and the traumas — of bringing news into a young, marketing-based company.

Justification

1. Access to the net keeps technical people happy. Considered this way, it's as much a perk as the coffee machine (and about as expensive to run — see below).

2. The net keeps us well informed of what's happening in the UNIX world, and, to a slightly lesser extent, in the industry at large. As a consultant, I find this knowledge increases my worth and the usefulness of my services to others, both inside and outside the company. And the non-computer-related postings broaden my mind!

3. As my company sells software products, it is very useful for us to learn of the good and bad experiences of existing users of software we already carry, which we may carry in the future, which is competitive to products we carry, or to which we may have to interface. Receiving this information from US sites, which tend to get product ahead of Europe, is an added bonus. Similar arguments undoubtedly hold for suppliers of computers and peripheral hardware.

4. My company runs what has come to be called a heterogeneous installation: we have many types of computer system from many different suppliers. Administration is a constant problem, as no standard tools yet exist for this task in such an environment. Sources and hints from the net have helped us on a number of occasions, the most recent being when the

1. Strictly speaking, we subscribe to **uknet**, the Britsh part of **eunet**, which, in turn, links to the US **Usenet**.

public domain *tar* was distributed late last year. We now use this program, which has compiled without problem for us in several environments, for cross-machine back-up as a matter of course. On its own, this program would justify many months of network charges.

5. The net is an educational tool: it can teach readers much about many aspects of programming.

Problems

1. You can spend a lot of time reading news — time when you ought, perhaps, to be doing other things. An intelligent browser, such as *rn*, is *de rigeur*. Even then, users have to spend the time to learn how it can best help them to cut down the volume of the news they see to a manageable level.
2. You can also spend a lot of time replying to news[2] (although surveys indicate that the vast majority of news readers are passive, and don't post followups).
3. Getting the news service running can be like searching for the light switch in a strange and totally dark room, involving a lot of groping around, and falling over things you didn't know were in your way. The situation getting better, however — see below.
4. News needs maintenance. I'd estimate that it costs us between two and four man-hours per week.
5. Good references and 'how-to' information are hard to find. Of course, user guides are posted to the net, but, firstly, you can't see them until the news service is running — the classic bootstrap problem; and secondly, they don't address the issues of administration and management at all. I can recommend two Nutshell handbooks published by O'Reilly and Associates: *Managing uucp and usenet* (ISBN 0-937175-09-9), and *Using uucp and usenet* (ISBN 0-937175-10-2). They are obtainable from specialist UNIX book stores[3].
6. News uses a lot of disk space, and quite a lot of CPU cycles and memory. The latter can be noticeable if several users are reading news at busy times of day. (The CPU costs of receiving news generally occur at night, and so don't cause problems.)
7. News is distributed on a 'best efforts' basis: failures at points throughout the network are commonplace, and can result in degradation or total loss of service for indefinite periods. To put it another way, the net is not a reliable medium. The level of service delivered would not be acceptable were it provided on a commercial basis. For a non-profit co-operative venture, it's what one might expect. The USA, until recently badly overloaded and disorganised, is getting a lot better as result of the commissioning of the central site, *uunet*. Europe, on the other hand, was running very smoothly until recently, but is beginning to creak a bit. (Plans are afoot to address these problems on short order.)

Practicalities

1. My company acts as a leaf node: we get our feed by polling Reading university (*reading*), which is a local telephone call away, at 1200 baud, and feed no other sites. The feed is compressed, giving a 'worthwhile' saving in transmission times, and, consequently, line charges.
2. We run news on an Olivetti/AT&T 3B2/400, and, following disk overflow problems on shared file systems, have dedicated a 17 MB file system to */usr/spool/news*. This is marginal: we keep the volume of news held on-line in check by not having a number of groups delivered to us, and by expiring a number of groups after one week rather than two. I'd estimate that 30 MB would be needed to have enough headroom to meet all eventualities. This is a lot of space on a comparatively small shared system. In addition, at least 2 MB of spool space are needed under */usr/spool/uucp* as temporary storage for incoming news. (For non-leaf sites, the situation is worse: they also need *uucp* spool space for outgoing files, which can quickly mount up if leaf sites fail to poll or respond to calls from their feed for any reason.)

2. For example, I spent a morning writing this article in response to a news posting...

3. And also (quick commercial) from Sphinx Ltd.

3. Apart from machine costs, which we do not budget, news costs us about $300 per quarter, split evenly between phone charges and our share of uknet's costs. These cash costs include some modem usage other than that required solely for news — in particular, UNIX mail, and cu and *kermit* dial-out connections to other sites. Uknet would charge us less if we provided primary news feeds to other sites. However, this would require us to dedicate to news more machine resources and more administration time than we do at present.

4. Getting news running was a real game, not least because it was all done in people's spare time, and not project-managed in any way. We obtained the source code from EUUG's 1986 public-domain software distribution tape (fine if you have a nine-track tape drive...), and compiled for our target by trial and error. (By the way, compiling *rn* was no problem at all: the program has one of the best installation scripts that I've ever seen.) Initial set-up was also a problem: the Nutshell books can help a lot here. Users of SCO XENIX and some other variants of UNIX may be able to get unsupported netnews binaries through the organisation which provides their operating system support. This should help quite a bit, but still leaves a fair amount of work to be done. All in all, I'd estimate the cost of getting our site reliably connected to others over dial-up links (including the reception of news) at approaching a man-month — decidedly non-trivial. This time could have been reduced had we not been so ambitious (for example, we spent a fair amount of time creating a script to drive a multi-standard smart modem bidirectionally).

5. Getting a news feed was quite easy, because I knew who to ask. Officially, you go to the central site in your country (note the bootstrap problem again), and they refer you to likely local feed sites, leaving you to take it from there. We know *uucp* well, and it took us about a week to bed down our connection to Reading. Again, the Nutshell books can take some of the pain out of this process.

6. The mailers delivered with systems are either dumb, or have a dumb default configuration. We've had to install a public-domain mailer (*smail*) on our 3B2 so as to handle domain addressing, and to make a reasonable attempt at handling the weird return addresses and paths that pop out of news postings. Even now, we're relying on *reading* and *ukc* to do most of the hard routing work. I anticipate any site that is serious about the news service and Usenet will find that the configuration job involves in similar work. (Happily, more and more systems are being delivered with *sendmail* these days, so you merely have to edit the configuration file.) (A touch of satire there...)

So, then, have the benefits of the news service outweighed the effort and expense of getting it running at our site?

I'd say that they have.

If you don't yet subscribe to news but want to, I hope this article has either given you some positive arguments to take to your management — or has spurred you towards presenting a *fait accompli* after you've got news up and running!

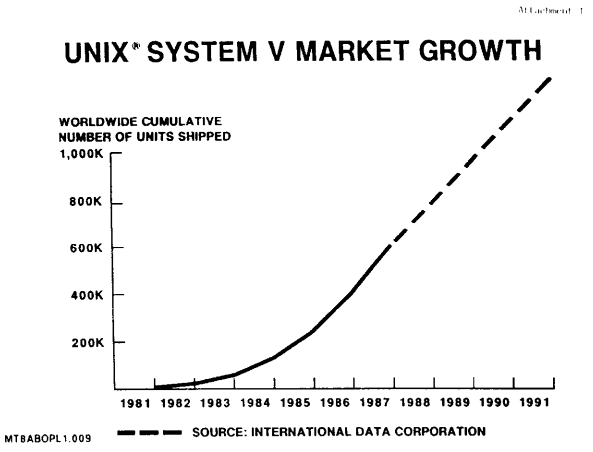
Unification and Openness

Janet Davis
janet@uel.uucp

AT&T UNIX Europe

AT&T announced plans for the next major release of UNIX System V and reaffirmed the openness of the operating system at UniForum in Dallas, Texas, in February of this year.

As statistics show more computers users are moving to UNIX Systems, In the last year the number of shipments of computers based on the UNIX Operating System increased by 60% and the International Data Corporation estimate that this growth will continue at an annual rate of 30% over the next three years.



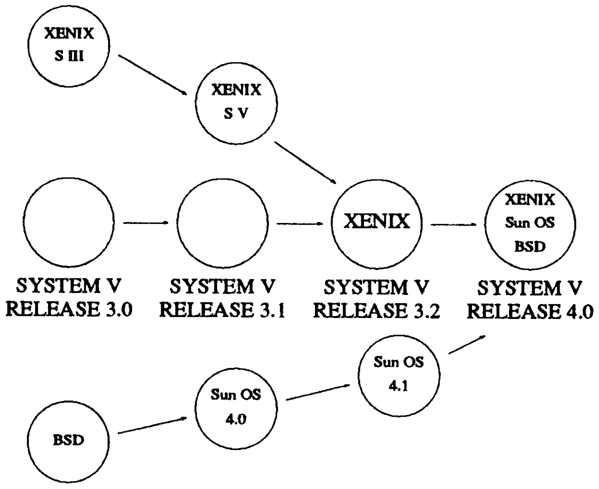
AT&T's plans for the next release of UNIX System V will support the growth of the UNIX Systems-based market. Two critical factors behind the success of UNIX Systems have been the move towards standardisation and the openness of UNIX System V.

AT&T's continuing commitment to both these areas will be backed up by a number of activities. AT&T will continue to provide both customers and the industry with information on the direction of UNIX System V. Seminars, such as the series planned for this summer covering UNIX System V Release 4.0, will be held on both technical and business issues. AT&T will also establish a UNIX System V OEM Licensee Group, open to all licensed commercial vendors. This group will meet regularly to share up-to-date information on UNIX System V developments and ideas for promoting the use of open systems. The aim behind all of these activities is to support the introduction of UNIX System V Release 4.0, and underscores AT&T's commitment to open systems.

Unification

Another major factor in the growth of the UNIX Systems market over the last year has been the move towards the unification of the major UNIX System derivatives, a move that will be of increasing importance over the next few years as the industry looks to a consolidation of the UNIX Systems market.

In 1987 the move towards unification occurred on two fronts. One being an alliance with Microsoft whereby XENIX System V compatibility would be consolidated into UNIX System V, the other being an alliance with Sun Microsystems whereby features of the Berkeley 4.2 and 4.3 systems (BSD) and Sun OS would be consolidated into UNIX System V.



a move that will be of increasing importance over the next few years as the industry looks to a consolidation of the UNIX Systems market.

In 1987 the move towards unification occurred on two fronts. One being an alliance with Microsoft whereby XENIX System V compatibility would be consolidated into UNIX System V, the other being an alliance with Sun Microsystems whereby features of the Berkeley 4.2 and 4.3 systems (BSD) and Sun OS would be consolidated into UNIX System V.

Application Binary Interface

As part of the strategy of unification AT&T have established a standard Application Binary Interface (ABI) for both the Intel 80386 and SPARC chip. Similar work is being carried out elsewhere for the Motorola 68000 chips. The ABI defines the binary interface just as the System V Interface Definition (SVID) defines the source code interface.

By providing application binary portability within an architecture, the ABI offers the type of portability normally associated only with PC applications which in turn will enlarge the market for software applications and preserve customers investments. Associated with the ABIs will be a Trademark Licensing Program allowing licensees to use the trademark UNIX to identify their sublicensed products.

Release 3.2

The first aspects of this unification work were seen at UniForum this year when AT&T demonstrated UNIX System V/386 Release 3.2. This is the first version of UNIX System V to incorporate XENIX System V compatibility and is expected to become the preferred multi-user, multi-tasking operating system for 80386-based computers. Running a wide selection of software, including software packages written for UNIX System V and XENIX System V for the Intel 80286 and 80386 chips, UNIX System V/386 Release 3.2 will also work with AT&T's Simul-Task software to allow users to run multiple MS DOS applications simultaneously. Licensing of UNIX System V, Release 3.2 will begin in mid-1988 with Release 3.2 for the 3B2 followed by the 80386 version. Sub-Licensees of the binary UNIX System V/386 Release 3.2 will have an option to license and use the trademark UNIX. Where the trademark is used the product must conform to AT&T's product and trademark usage specifications. For end-users this will mean that when they purchase an 80386-based system running UNIX System V they will be assured of the binary portability of applications.

Release 4.0

Release 4.0, the next major release of UNIX System V, will broaden the market for UNIX System software by further unifying the major derivatives of the UNIX Operating System. As well as incorporating features of XENIX and BSD and Sun OS, new features such as real-time capabilities, improvements to systems operations, administration and management, enhanced networking and features designed for international markets will be included.

The merging of BSD and Sun OS features into UNIX System V will take place in three phases. Phase 1 will see Sun OS 4.1 developed to comply with the SVID. In Phase 2 AT&T will merge Berkeley features into UNIX System V, Release 4.0 and add new AT&T features. This version will comply with the IEEE POSIX standard and will be consistent with the operating system specifications of X/Open. The third and final phase will see the restructuring of the kernel.

BSD and Sun OS features widely used in the industry will be incorporated into Release 4.0 so that users familiar with these features will find it easy to migrate to a single common UNIX System V base. Along with RFS, NFS will provide distributed access and operations on UNIX System file system objects. RFS and NFS will each be supported as an independent package with an administrative interface that will give an administrator a single, consistent view of distributed file access. Remote Procedure Call (RPC), a tool for developing distributed processing applications for a variety of manufacturers' hardware will also be incorporated. RPC lets a process executing on one machine issue service requests to a process on another machine.

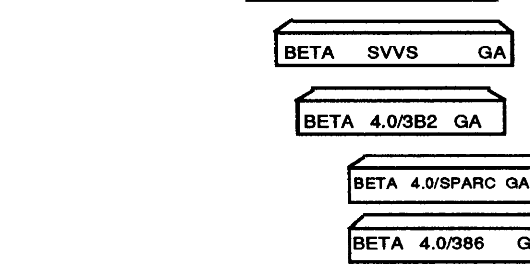
Significant enhancements are also planned in the areas of system operations, administration and maintenance will include backup and restore, configuration management software installation and distribution, message handling facility and remote operation. Real time enhancements include a scheduler, general events mechanism and asynchronous I/O while internationalisation features such as a C compiler that conforms to the ANSI X3J11 international standards and a message handling facility will be included.

Conformance to Industry Standards

With Release 4.0 AT&T reaffirms its commitment to supporting industry standards. Release 4.0 will conform to the forthcoming IEEE POSIX standard, which is due to be published and adopted in 1988. Release 4.0 will also be consistent with the operating system specifications of X/Open in line with AT&T's commitment of providing the industry with products that are compatible with the X/Open Common Application Environment.

When Will All This Happen?

In preparation for the introduction of Release 4.0, AT&T will be holding briefing seminars on Release 4.0 for computer manufacturers and software developers during 1988.



This year will also see the OEM Licensee Group established to set up a channel of communication and input on the development and direction of UNIX System V. As well as preparing much of the groundwork for Release 4.0, AT&T will bring UNIX System V Release 3.2 to the market in 1988. This release will first be delivered on the 3B2 and then on the 80386 chip. 1988 will also see work commence on a draft update of the ABI for the SPARC chip, work which will be completed some time in 1989. During 1989 AT&T will update the SVID and prepare a version of the System V Verification Suite for beta release to customers wanting to test their systems conformance to the SVID. Finally the move towards the unification of UNIX Systems will culminate in 1989 when beta versions of Release 4.0 will be made available first for the 3B2 followed by the 80386 and SPARC chips.

UNIX System V has been successful in the marketplace because of its consistency, openness, widespread availability and powerful capabilities. The next two years will see AT&T expand its open systems policy and continue to develop and enhance UNIX System V.

OPEN LOOK

AT&T have announced the OPEN LOOK user interface, it employs commonsense graphic symbols instead of written commands to help users work more efficiently with their UNIX System V-based computers.

"OPEN LOOK will change the way the industry thinks of the UNIX system," said Vittorio Cassoni, president of AT&T's Data Systems Group. "This interface brings the benefits of the UNIX system to a whole new group of users who otherwise might never have taken advantage of the power of a UNIX system-based computer."

The OPEN LOOK technology was designed for AT&T by Sun Microsystems Inc. of Mountain View, Calif. Sun's design is based on original work, contributions from AT&T, and on technology licensed from Xerox Corporation, which originated many of the concepts present in today's computer interfaces.

The OPEN LOOK interface's graphic symbols include push pins to 'pin' important menus to the screen for further reference and an elevator to move up or down in the text. To print or store files, users move a hand-held mouse to push labeled buttons designed to look like those on a household appliance.

As the name implies the OPEN LOOK user interface supports AT&T's commitment to open systems and the need for a standard user interface. Scott McNealy, president of Sun Microsystems, says that it represents the next critical step in truly expanding the UNIX marketplace. Applications developed with the OPEN LOOK interface can vie for a larger market because the interface is standard.

The interface has already generated endorsements from key computer system suppliers, PC and workstation software suppliers and system suppliers.

OPEN LOOK combined with ABI, will provide a common application platform which has, until now, only been found on MS-DOS machines. Because of this OPEN LOOK has been endorsed by some of the big names in the PC world, these include: Lotus, Ashton-Tate, and Xerox.

In addition to being easy for them to learn, the OPEN LOOK interface will make users more productive because it allows them to create multiple 'windows' on their computer screens, each of which can perform a different task simultaneously.

Programmers will find that the various Application Programmer Interface (API) Toolkits AT&T plans to release will give them a set of tools — or pre-programmed components — to make it more efficient to write new applications by reducing the amount of code that needs to be written per function. As OPEN LOOK will have a defined program interface, software portability will be increased.

AT&T will circulate OPEN LOOK specifications for comment this summer and will make them available in the third quarter of this year. These will include a specification of the common style for applications — The Applications Style Guide — as well as descriptions of the programming interface for OPEN LOOK under two toolkits, both of which AT&T will support via a single graphics system platform. They are the XT toolkit based on the X Windows and the NDE toolkit based on NeWS.

The first availability of OPEN LOOK features in an AT&T product will be this summer in a window manager for the 6386 workstation, followed by an XT toolkit in the first quarter 1989.

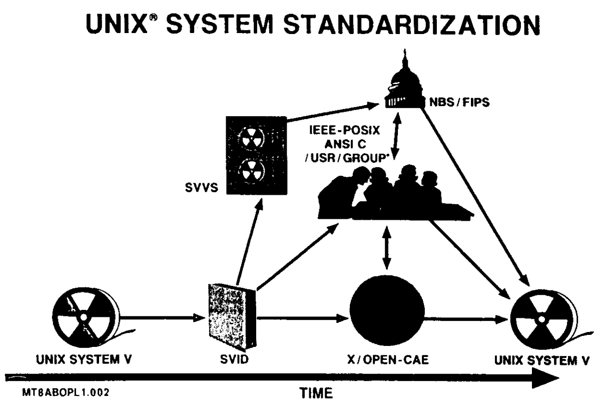
In keeping with its commitment to support standards, AT&T said that as they become accepted, the company would support API's for emerging standard interfaces. AT&T also will license source code for the various toolkits supporting the OPEN LOOK user interface.

The OPEN LOOK user interface toolkits are scheduled to be available in source form in early 1989.

The OPEN LOOK user interface is designed to be useful into the 1990's. For instance, unlike some graphical interfaces, the OPEN LOOK interface is designed for a wide range of applications from simple document processing to much more sophisticated computer-aided engineering (CAE). In addition, the graphics perform well whether they appear on a PC or a high resolution engineering workstation. Also, the interface will support a variety of terminals accessing different applications.

AT&T today also announced it will co-sponsor with Sun Microsystem a series of eight, three-day conferences around the world beginning in September to give independent software vendors, value-added resellers and large corporate users a preview of the key technical features of UNIX System V Release 4.0, including the newly announced OPEN LOOK interface.

Attachment 2



Book Review: 2 Books on ANSI C (Draft)

Reviewed by
Andrew Macpherson
andrew@stl.stc.co.uk

STC Technology Ltd.

The ANSI Draft standard has been out for public comment for almost a year, and we are now seeing the books which will guide us through the exact formal definitions to a working appreciation of where the language has gone.

There is an attitude which says that once a language is set in a standard it becomes useless. C shows little signs of this failing, but those who subscribe to this viewpoint can take heart: in both cases C++ was used to check the code!

The C Book

Featuring the draft ANSI C Standard.
Mike Banahan, The Instruction Set Series — Addison Wesley, 1988, ISBN 0–201–17370–0. (U.K.) price 15.95, Soft Back, 268 pp, Size 23.5 cm x 15.5 cm.

To many the acronym X3J11 is just so much gibberish. To the C programming community however, the J11 committee of ANSI's X3 secretariat is a medium term source of fairly fundamental change to the language. Mike Banahan's early involvement in that committee, and his well-known skills as a communicator, make him an ideal candidate to present the changes and the rationale for some of the apparent inconsistencies introduced or addressed by the Draft Standard.

Not content with this, *The C Book* is more than just a catalogue of changes and how they will affect the practice of programming and porting C. Its scope also encompasses a balanced tutorial on C for practising programmers who wish to become familiar with the language.

The format is very much an informal tutorial style — the difficult concepts are reserved for the later stages, and each new concept has an illustrative fragment of code to demonstrate its use. In fact the code fragments are usually complete working programs, all of which have been tested from the text (with one exception, noted in the text on page 207). Significantly, Mike recognises that some of the working of C is best reserved until one has experience of the language, and recommends that portions of the book be skipped, or at least skimmed, on first reading, and only returned to when the reader has six months familiarity with the language. Each chapter has, at the end, a review exercise to test the reader's understanding and full answers are given at the end of the book.

It works well. The first six chapters deal with the language *per se*, with only a few difficult points reserved for chapter 8. The style is easy to follow, and the occasional wry wit helps make points while improving the readability. Chapter 7, the pre-processor, becomes clear on a second reading, but I would advise skipping past the informal discussion of tokens at the bottom of page 158 as it tends to cloud the issue, only coming back to it if you have to. The abuse of the second language — the pre-processor — is well warned against:

"The urge to maim the author of a piece of code becomes very strong when you suddenly come across

```
#else
        }
#endif
```

with no `#if` or whatever immediately visible above."

Also don't be confused by the missing `__DATE__`, on page 166, it is explained on the opposite page.

Chapter 8 is a discussion of many things which are not, in general, needed to make use of the language, in particular the new and unfamiliar storage qualifiers `const` and `volatile`, along with a discussion of linkage, sequence points (where side-effects are resolved) and `typedefs`. This last is put in the realm of system header files, rather than a useful tool to the application programmer's hand, and may reflect, in part, the same historical perspective that led to the explanation that `enums` did not exist when Mike learned the language.

The discussion of the standard libraries is divided by function (or header file), and quickly establishes that the UNIX interface has not been taken up *in toto*. Goodbye `creat`, goodbye `unlink`, hello `delete`. I expect this to be the most useful section of the book for me in the long-term.

This is not a book for the computer novice. It does not pretend to be. It is certainly suitable as a course-book both for home-study and formal instruction. I would rate it as essential reading for someone contemplating their first C language project, who needs to know what C will and will not do for them, the limits of the automatic error detection at compile time, and what is left to the programmer.

The C Programming Language

Second Edition, Based on Draft-Proposed ANSI C. Brian W Kernighan and Dennis M Ritchie, Prentice-Hall Software Series, 1988, ISBN 0–13–110362–8. (U.K.) Price 24.95, Soft Back, 272 pp, Size 23.5 cm x 17.7 cm.
Available — Real Soon Now

The White Book, K&R, the C programmer's bible — a book so deeply engrained into one's working environment for the past ten years that it is nigh on impossible to look objectively at it, and here it is in a revised second edition with the emphasis shifted to what may be.

This is not a new book, but it will be a best-seller in the UNIX community. It is a rewrite of the 1978 C book, with new examples, a C–declaration translator `dcl`, a new chapter on the ANSI standard libraries, and the benefit of improved diagrams (`pic`).

Saying it like that gives the wrong impression, however. The first edition of this book has for so long been the standard for the language, a sufficiently usable standard that compilers could be written and tested against the text, that every working C programmer must have had a copy. This second edition is a major revision of text, order, examples; it gives credibility to the ANSI standard.

How then does the new book differ from the old? Other than the new typesetting, and the unacknowledged font used in chapter and section headings, the text has been thoroughly revised. No more is it a tutorial and reference manual for C, rather a tutorial and reference manual for ANSI Standard C. All the new features that have been added to the language are dealt with, and considerable emphasis is placed on the standard libraries.

For those who want a quick introduction to the new language, Appendix C is a summary of the changes introduced by the standard, while Appendix A is the language reference manual. The reference manual contains the bones of a `yacc` parser for the new grammar, but otherwise follows the format of the first edition. Appendix B deals with the standard library, it is equivalent to chapter 9 of *The C Book*, with slightly less explanatory text, but is a more useful quick reference.

While usable as a study document, this, like the previous edition, is a working programmers book. Anyone seriously using ANSI C, particularly on UNIX, should have a copy.

Book Review: UNIX Products for the Office

Reviewed by
Tony Bamford
afb@phcomp.co.uk

Parliament Hill Computers Ltd.

UNIX Products for the Office

The National Centre for Information Technology, Published by The National Computing Centre Ltd., 1988, ISBN 0-85012-701-7. Price (UK) 45, Soft back, 266 pp.

Its all very well knowing where your next GNU EMACS update is coming from but what do you do when you want a prospect tracking system to run on your UNIX box? Never fear, NCC have the answer.

UNIX Products for the Office is a catalogue of UNIX products available within the UK that are designed and intended specifically for the office environment.

It is divided into 4 main parts, an introduction, a detailed description of products, an index by description and an index by supplier.

The introduction deals with what UNIX is, why UNIX is wonderful for office applications and how to choose a UNIX system if you don't have one already. The product description forms the major part of the handbook and consists of about 200 pages, each product getting half an A4 page. The half page of information gives a few lines describing what the product does, a list of applications for which the product is suitable, a list of users the product addresses and some details of the type of hardware needed to run it. Then come financial matters, how much it costs to buy or, where applicable, rent and how much annual maintenance costs. Finally a UK supplier of the product is named.

This book is a great idea, a guide to UNIX products in the marketplace is long overdue.

However, there are some problems with the catalogue, a fair proportion of the entries don't mention prices. Also, only one supplier is listed per product, this is to be expected for software written by small software houses, but one UK supplier for INFORMIX? Other problems appear when the catalogue is used and suppliers say "sorry, we don't deal with that product any more".

The key to the problem is outlined in the introduction to the catalogue, which states:

> Any volume of this type is absolutely dependent on the good will of the product suppliers in completing the questionnaires ...

This implies that many suppliers returned half-completed questionnaires. Thus, we can only hope that the book becomes popular and suppliers are encouraged to provide the NCC with complete information.

So my conclusion is good as far as it goes, but the second edition should be more useful.

EUUG Conference Proceedings

Here are the abstracts of the papers delivered at the EUUG 10th Anniversary Conference (Part II) in London this April. Please contact the authors if you would like a copy of a paper.

Thanks are due to Stuart Mc Robert (sm@doc.ic.ac.uk) who also typeset the proceedings.

OFS — an Optical View of a UNIX File System

Paulo Amaral

GIPSI-SM90[1]
c/o INRIA
BP105
78153 LE CHESNAY CEDEX
FRANCE
mcvax!inria!gipsi!paulo

Abstract

The design and implementation of the Optical File System (OFS) is described. It was conceived to run under UNIX and to deal with Optical Disks. We explain our view on how to develop a file system, at UNIX user level, with a WORM (Write Once—Read Many) device. OFS manipulates multiple file versions automatically. It also works upon an implementation of atomic transactions: fault tolerance implications are studied. Finally, we describe our experience using the OFS by means of a backup utility, that has been used by our software research team since October 1987.

Software Re-engineering using C++

Bruce Anderson
Sanjiv Gossain

Electronic Systems Engineering
University of Essex
bruce@ese.essex.ac.uk,
goss@ese.essex.ac.uk

The plan for our experiment was to take a piece of software and rewrite it in C++. We wanted the program in question to be locally-generated, written in C, widely used and to be a generic program, one which was typical of a class of programs that were either actually written or likely to be needed. Our idea was to proceed in small steps and to reflect on each step.

Measuring File System Activity in the UNIX System

Maurice J. Bach[2]
Ron Gomes

AT&T Information Systems
190 River Road, Summit, NJ 07901

Abstract

We describe an analysis of system call activity (particularly file system activity) made on several UNIX systems in a software development lab. The measurements were motivated by design work in support of distributed file systems; the intent was to characterise such things as system call frequencies, file system access patterns, and caching behaviour, and to identify performance bottlenecks which might have been missed by existing measurement tools. Among the more important results of the study:

- Most system call activity is file system activity.
- Most *reads* and *writes* are not matched to the file system block size.
- Most terminal I/O is done a single character at a time
- Operations on directories dominate buffer cache activity even though only a small part of the cache contains directory data.
- Most buffer cache hits result from repeated access by a single process.

A UNIX Environment for the GOTHIC Kernel

Pascale Le Certen,
Bèeatrice Michel,
Bull Recherche.

Gilles Muller,
IRISA-INRIA
Campus de Beaulieu, 35042 Rennes-CEDEX

Creating an operating system on an open machine, implies the use of development methods. This report describes the way chosen to implement the kernel of the GOTHIC distributed system and a well suited environment. Our goal is to design a development system which can run on the successive versions of the kernel. The major advantage of that method is to intensively test the kernel for programming and design errors.

UNIX Around the World

Sunil K Das

City University London,
Computer Science Department
London EC1V 0HB, UK
sunil@cs.city.ac.uk

Abstract

In the Preface to the Eighth and Ninth Editions of the Programmer's Manual for the UNIX Time-Sharing System, Doug McIlroy says that the volumes describe the lineal descent of the original operating system pioneered by Ken Thompson and Dennis Ritchie. Distributed computing proved to be the distinctive theme of the landmark Eighth Edition: Dennis Ritchie's coroutine-based stream IO system, and the Datakit virtual circuit switch realisation by Lee McMahon and Bill Marshall, provided the basis for networking, Peter Weinberger's remote file systems made it painless, and Rob Pike's software for the Teletype 5620 moved system action

1. GIPSI-SM90 is sponsored by the French Ministry of Research and Technology under the contracts 83-B1032 84-E0651 85-B0524

2. Author's current address: IBM Israel Scientific Center, Technion City, Haifa, Israel.

right out to the terminal.

Users distributed around the world is the theme of the Spring 1988 EUUG Conference. The Conference Chairman discusses here why users around the world have demanded to use UNIX, why UNIX has proved successful around the world, and the future of the UNIX system in the world marketplace.

The paper finishes with a citation of the original and innovatory contributions made by many of the speakers who travelled from all over the world to be at the the EUUG's Conference held at the Queen Elizabeth II Conference Centre, London in April 1988.

UNO: USENET News on Optical Disk

A. Garibbo,
L. Regoli,
G. Succi

University of Genoa
Italy

Introduction

The size of a WORM optical disk is greater than a Gbyte and it is likely to grow fast within few years; moreover storing and retrieving USENET news is becoming tedious and difficult: at present time a user has easy access to news if he knows exactly which ones he wants to consult; besides reading daily news takes little time using standard read-news tools.

Troubles arise when one wants to find some news only knowing few features because the help he has is merely a hierarchical organisation of the news supplied by the USENET system: actually, such a tree-shaped framework seems to be quite unsuitable as long as:

i. the structure is not strongly enforced

ii. quite different leaves lie in the same directory

Owing to the high rate of news traffic, lots of space is needed, and usually each local network connected with USENET either devotes too much space to archiving or it needs frequent backup on tape.

UNO – USENET News on Optical disk – attempts to solve this kind of problems, since more than four years of full news, at the present rate, can be archived on a WORM disk.

All facilities provided by standard readnews tools are enclosed in UNO; moreover it supports an incremental knowledge driven search, which allows interactive data retrieving without either knowing exactly the wanted news or having to deal with all the news of a USENET directory.

UNO provides easy interaction through a smart query language, remote query and intelligent programmable selection of relevant news.

UNO was developed an a workstation named Arianna, based on a National 32032 processor, which runs UNIX System V.3. A WORM disk is fully integrated in the global file system; UNO is designed in C++ according to object oriented programming and software engineering criteria.

Evolution of the SunOS Programming Environment

Robert A. Gingell

Sun Microsystems, Inc.
2550 Garcia Ave.
Mountain View, CA 94043 USA

Abstract

Recent changes to Sun's implementation of the UNIX operating system (SunOS) have provided new functionality, primarily file mapping and shared libraries. These capabilities, and the mechanisms used to build them, have made significant changes to the programming environment the system offers. Assimilating these new facilities presents many opportunities and challenges to the application programmer, and these are explored in this paper.

The new mechanisms also provide the application programmer with a flexibility comparable to that previously reserved for the operating system developer. Much of this flexibility is based on mechanisms for dynamic linking that support interposition. The future developments and ramifications of these mechanisms, as well as other areas for similar system refinements, are also explored.

Multiprocessor UNIX: Separate Processing of I/O

A.J. van de Goor,
Delft University of Technology,
Department of Electrical Engineering,
Mekelweg 4,
P.O. Box 5031,
2600 GA Delft,
The Netherlands.
vdgoor@dutesta.UUCP

A. Moolenaar,
Oce Nederland B.V.,
St. Urbanusweg 126,
P.O. Box 101,
5900 MA Venlo,
The Netherlands.
mool@oce.nl.UUCP

J.M. Mulder,
Delft University of Technology,
hansm@dutesta.UUCP

Abstract

Making UNIX suitable for a multiprocessor system is a logical step because of the wide acceptance of UNIX and the decreasing cost of hardware. The multiprocessor adaptation, however, is not trivial because of some of the assumptions the UNIX kernel is based on. This paper illustrates, on a high level, the performance considerations which guided the design of a UNIX multiprocessor, and it describes specifically the modifications required to implement the I/O kernel layers on dedicated I/O processors. This implementation was based on the concepts of horizontal and vertical data sharing.

System V Release 3, Diskless Workstations and NFS

Robert Cranmer-Gordon,
Bill Fraser-Campbell,
Mike Kelly,
Peter Tyrrell

The Instruction Set
rob@inset.co.uk,
bill@inset.co.uk,

wot@inset.co.uk,
petet@inset.co.uk

Abstract
Diskless UNIX workstations are becoming a fashionable way of providing users with high levels of facilities and performance at low cost. To date, most implementations of UNIX for diskless computers have been based on 4.2BSD. This paper describes some major modifications made to Motorola System V/68 to produce a version of System V Release 3 capable of supporting diskless machines.

To make diskless operation possible using Sun's Network File System (NFS over Ethernet, the File System Switch feature of System V Release 3 has been replaced with the Sun Virtual File System (VFS) switching arrangement. However, the Release 3 STREAMS architecture has been retained as a framework for Internet protocol software to permit a (comparatively) easy switch to OSI protocols in the future. The BOOTP (RFC 951) and TFTP protocols are used for bootstrapping diskless machines.

The paper presents details of the method used to marry NFS and STREAMS, performance enhancements to the Sun distributed record locking and experiences with BOOTP. It also lists those awkward places where the requirements of NFS and the System V Interface Definition (SVID) conflict.

Implementation of X.25 PLP in ISO 8802 LAN Environments

S.A. Hussain,
J. Ølnes,
T. Grimstad

Norsk Regnesentral,
Blindern
0314 Oslo 3
anwar%vax.nr.uninett@tor.nta.no

Abstract
The X.25 Packet Layer (ISO 8208) and Class II of LLC (ISO 8802/2) are both implemented in the kernel of Berkeley UNIX 4.2BSD on a VAX 11/750 as a new communication domain (AF_XLAN). It is accessible using the IPC primitives provided by 4.2BSD. X.25 PLP's stream services are accessible via stream sockets. Class II of the LLC's datagram services are accessible via raw datagram sockets and stream services via raw stream sockets.

General Purpose Transaction Support Features for the UNIX Operating System

S. G. Marcie
R. L. Holt

NCR Corporation
E&M Columbia
W. Columbia, South Carolina 29169

Abstract
This paper describes the features of NCR's General Purpose Transaction Facility (GPTF), an extension to NCR's implement-ation of UNIX System V for the TOWER supermicrocomputer. Timer signals with millisecond resolution are presented. Performance of process synchronisation and interprocess communication is improved via a set of semaphore primitives which executes in the user program environment and operates on structures which exist in standard UNIX System V shared memory. A scheduler is described which reduces process switching latency and provides process scheduling among both realtime and timesharing priority classes.

Additionally, a mechanism is provided to lock a process in memory so that it is immune to paging. Scheduling latency is reduced through voluntary preemption within the kernel. A novel disk I/O scheduler provides the ability to schedule disk requests according to process priority, seek distance, or some configurable combination of both parameters.

User access to the transaction processing facilities is provided via a set of system calls and shell commands. A user friendly interface is provided to allow a superuser to control such access.

Grep Wars

Andrew Hume

AT&T Bell Laboratories
Murray Hill, New Jersey 07974
research!andrew

Abstract
Subsequent to the Sixth Edition of the UNIX system there have been different versions of the searching tool *grep* using different algorithms tuned for different types of search patterns. Friendly competition between the tools has yielded a succession of performance enhancements.

We describe the latest round of improvements, based on the *fio* fast I/O library and incorporating the Boyer-Moore algorithm. Although *grep* is now 3–4 times faster than it was, *egrep* is now typically 8–10 (for some common patterns 30–40) times faster than the new *grep*.

Yacc Meets C++

Stephen C. Johnson

Ardent Computer Corp.[3]
880 W. Maude Ave.
Sunnyvale, CA, USA, 94086

Abstract
The fundamental notion of attribute grammars is that values are associated with the components of a grammar rule; these values may be computed by *synthesising* the values of the left component from those of the right components, or *inheriting* the values of the right components from those of the left component.

The yacc parser generator, in use for over 15 years, allows attributes to be synthesised; in fact, arbitrary segments of code can be executed as parsing takes place. For the last decade, yacc has supported arbitrary data types as synthesised values and performed type checking on these synthesised values. It is natural to think of this synthesis as associating a value of a particular type to a grammar symbol when a grammar rule deriving that symbol is recognised.

Languages such as C++ support abstract data types that permit functions as well as values to be associated with objects of a given type. In this framework, it appears natural to extend the

3. Much of this work was done when the author was employed by AT&T Information Systems

idea of computing a value at a grammar rule to that of defining a function at a rule. The definition of the function for a given object of a given type depends on the rule used to construct that object.

In fact, this notion can be used to generalise both inherited and synthesised attributes, unifying them and allowing even more expressive power.

This paper explores these notions, and shows how this rule-based definition of functions allows for easier definitions and much more flexibility in some cases. Several examples are given that are hard to express using traditional techniques, but are naturally expressed using this framework.

Software Tools for Music - or - Communications Standard Works!

David Keeffe

Siemens Ltd., Systems Development Group,
Woodley, Reading, UK
ukc!siesoft!dk

Introduction

Described here is the evolution of a small suite of programs for the composition and performance of music. They started life as a personal interest, inspired in part by Peter Langston's work [Langston 86]. As they developed, however, a use for the programs was seen as an unusual and illustrative aid for exhibiting computing equipment.

At the Systems Development Group, we are involved variously in developing systems which address the problems of UNIX and DOS communication, in developing software for a graphics workstation, and generally in improving the flexibility and usability of Siemens range of UNIX and DOS machines. Would it not then be a good idea if an 'earcatching' package could be built which combined all this?

As such a system is to receive close scrutiny, the musical ideas must have a reasonable foundation: the computer should not be seen as simply a glorified tape machine.

The aim of this paper, then, is to present several facets of the music system: of course, the musical ideas are central, but there are also other lessons to be learnt. There are two central foundations of the design of the system: the first is the Musical Instrument Digital Interface, or MIDI: there will not be much said about it, as there isn't much to say — the standard itself is only about one-third the length of this paper! What should become clear is not only the way MIDI allows the system to function but also how such a useful standard can make writing other music programs so much easier and more effective. The second foundation is the legacy of the UNIX operating system: it is that legacy which makes the whole thing fit together.

Why music? The composition of music is generally thought of as one of the most abstract of human activities. While not ever hoping to replace the human musician, the computer can be used to experiment with music, as well as providing a base for a diverting exercise in analysis and programming. Also, computer music is strangely attractive, like high-quality intelligent speech systems.

An Overview of the Gothix Distributed System

Alain Kermarrec

IRISA – Campus de Beaulieu –
35042 RENNES-CEDEX - FRANCE –
kermarre@irisa.irisa.fr

Introduction

Currently under development at the IRISA/INRIA, GOTHIC is intended to be an integrated distributed system implemented on a network of multi-processor machine BULL SPS7. Since the development of the GOTHIC kernel is assumed to take a rather long time, it was decided to build on UNIX machines (a Network of SUN running under UNIX 4.2BSD) a system which provides the same interface as GOTHIC in order to start the development of applications. The first release of this system called GOTHIX is currently under test. This paper first describes the concepts developed in both systems and then discusses some implementation details of GOTHIX.

A Tool-based 3-D Modelling and Animation Workstation

Samuel J. Leffler
Eben F. Ostby
William T. Reeves

Animation Research and Development Group
Pixar
3240 Kerner Blvd.
San Rafael, CA. 94901

Abstract

A tool-based system for 3-D modelling and animation is presented. Each *tool* is a separate program that operates as an independent UNIX process. Tools utilise a window-oriented display package, an event-based input system, and a large graphics database that resides in shared memory in providing interactive and non-interactive functions. The system described here is being developed for use in the production of 3-D animated sequences and as a testbed for research in 3-D modelling and animation. The architecture of the system and the motivation behind the tool-based approach is described.

Word Manipulation in Online Catalog Searching: Using the UNIX System for Library Experiments

Michael Lesk

Department of Computer Science
University College London
Gower St
London WC1E 6BT

Bellcore
435 South St
Morristown, NJ 07960

Abstract

Online public access catalogs are often plagued with very short queries and very short document descriptions. As a result performance may be poor and the users are dissatisfied. To improve recall, in particular to deal with query terms not found in the collection, a machine-readable dictionary can be used to identify related terms by overlap of defining words. To improve precision, phrases can be retrieved and the user asked to pick the appropriate ones. A demonstration system is running on 72,000 records from the British Library Eighteenth Century Short Title Catalog.

A UNIX system is a good way to implement this software, because of its advantages of easy programming, availability on small machines, and advanced data base routines.

Help! I'm Losing My Files!

John Lions

University of New South Wales
Kensington 2033
Australia

Abstract

Managing large collections of miscellaneous files can present a problem for individual users of a UNIX system. Keeping track of files that are still wanted and useful, finding and eliminating files that are no longer needed, and reorganising the file hierarchy from time to time may not be trivial if the set of files is large. Outlines are drawn for a partial solution involving index files, embedded keyword lists, a procedure for revising file pathnames and the implementation of a daemon secretary to keep everything tidy.

A Toolkit for Software Configuration Management

Axel Mahler,
Andreas Lampen

Technische Universität Berlin

Abstract

For almost ten years, `make` has been a most important tool for development and maintenance of software systems. Its general usefulness and the simple formalism of the `makefile` made `make` one of the most popular UNIX tools. However, with the increased upcoming of software production environments, there is a growing awareness for the matter of *software configuration management* which unveiled a number of shortcomings of `make`. Particularly the lack of support for version control and project organisation imposed a hard limit on the suitability of `make` for more complex development and maintenance applications.

Recently, several programs have been developed to tackle some of the problems not sufficiently solved by `make`. `Shape`, the system described in this paper, integrates a sophisticated version control system with a significantly improved `make` functionality, while retaining full upward compatibility with `makefiles`. `Shape`'s procedure of identifying appropriate component versions that together form a meaningful system configuration, may be completely controlled by user-supplied *configuration selection rules*. Selection rules are placed in the `shapefile`, `shape`'s counterpart to the `makefile`.

The `shape` system consists of commands for version control and the `shape` program itself. It is implemented on top of the *Attribute File System* (AFS) interface. The AFS is an abstraction from an underlying data storage facility, such as the UNIX filesystem. The AFS allows to attach any number of attributes to document instances (e.g., one particular version) and to retrieve them by specifying a set of desired attributes rather than giving just a (path–) name. This approach gives an application transparent access to all instances of a document without the need to know anything about their representation. So, it is also possible to employ different data storage facilities, as for instance dedicated software engineering databases.

The project organisation scheme of `shape` provides support for small (one man), medium, and large projects (multiple programmers/workstation network).

Design of and Experience with a Software Documentation Tool

José A. Mañas
Tomás de Miguel

Dept. Ingeniería Telemática
E.T.S.I. Telecomunicación
Ciudad Universitaria
E-28040 MADRID
SPAIN
jmanas@goya.uucp
tmiguel@goya.uucp

Abstract

A UNIX tool is presented that permits to write documented code in a text oriented fashion, looking for humans that have to read, understand, and maintain it, rather than thinking for language processors that have to compile it. The tool permits handling any text processing system, as well as any target language. Several files may be documented and maintained as a single unit, thus helping in keeping them coherent. The tool may easily and productively interact with standard UNIX tools. The design criteria, basic features, and some real examples of utilisation are presented.

UNIX Past, Present, and Future: Changing Roles, Changing Technologies

John R. Mashey

MIPS Computer Systems
Sunnyvale, CA 94086

Introduction

The UNIX operating system seems to defy the laws of physics by remaining in perpetual motion. This paper takes a brief look at where it's been, where it is, and where it might be going. In particular, UNIX stands as a major beneficiary of the the current developments in RISC microprocessors.

Multilevel Security with Fewer Fetters

M. D. McIlroy
J. A. Reeds

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Abstract

We have built an experimental UNIX system that provides security labels (document classifications), where the security labels are calculated dynamically at the granularity of kernel activity, namely, at each data transfer between files and processes. Labels follow data through the system and maintain the lowest possible classification level consistent with the requirement that the labels of outputs dominate the labels of inputs from which they were computed. More rigid control is exerted over the labels of data passing out of reach of the system to and from tapes, communication lines, terminals, and the like. Necessary exceptions to the security rules (as for system administration, user authentication, or document declassification) are handled by a simple, but general, privilege mechanism that can restrict the exceptions to trusted programs run by 'licensed' users. Privileges are subdivided; there is no omnipotent superuser. Carefully arranged data structures and checking algorithms accomplish this fine-grained security

control at a cost of only a few percent in running time.

Dynamic labels should help mitigate the suffocating tendencies of multilevel security. At the same time dynamic labels admit covert channels by which dishonest, but authorised, users can leak data to unauthorised places at modest rates. The system is still highly resistant to other kinds of threat: intrusion, corruption of data by unauthorised users, Trojan horses, administrative mistakes, and joyriding superusers. In most real settings, we believe, worries about potential leaks will be far outweighed by these latter concerns and by the overriding consideration of utility.

Directly Mapped Files

Andreas Meyer

Stollmann GmbH,
Max Brauer Allee 81
D-2000 Hamburg 50
West Germany

Abstract

Directly Mapped Files is a file access method implemented under UNIX System V Release 3.

The entire file appears to the user process like a large byte array in the virtual address space and may be accessed without any read, write or seek operations. Thus, many programs, especially those working on data bases, intermediate files, or complex data structures, may be written much more easily and run faster. The paper describes the implementation in the UNIX kernel and its direct relationship to the demand paging algorithms. An example (*ar*) demonstrates the issues for user programs.

SunOS Virtual Memory Implementation

Joseph P. Moran

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043 USA

Abstract

The implementation of a new virtual memory (VM) system for Sun's implementation of the UNIX operating system (SunOS[4]) is described. The new VM system was designed for extensibility and portability using an object-oriented design carefully constructed to not compromise efficiency. The basic implementation abstractions of the new VM system and how they are managed are described. Some of the more interesting problems encountered with a system based on mapped objects and the resolution taken to these problems are described.

Adventures in UNIX Arithmetic

Robert Morris

National Computer Security Center
Fort George Meade
Maryland 20755
USA
RMorris@dockmaster.arpa

4. SunOS is a trademark of Sun Microsystems.

The problems of writing mathematical software under UNIX are described. This comes about as the accuracy of standard mathematical functions (e.g., *sin*, or *log* is something rarely considered important by the supplier of a computer system, the *guru* that does the port will probably comment that 'floating point is for users' and forget this important issue.

This paper is a plea to manufacturers to provide something that works to the accuracy that the hardware can support.

The JUNET Environment

Jun Murai

Computer Centre
University of Tokyo
Japan
jun@u-tokyo.junet

Abstract

The JUNET environment consists of various Kanjified public domain utilities and some original tools providing Kanji capabilities. The design and implementation of this environment will be described, as well as the current status of JUNET itself.

POSIX — A Standard Interface

Jim R Oldroyd

The Instruction Set
jr@inset.co.uk

Abstract

There is a bewilderingly large number of UNIX systems in existence today. Most are derived from one of two main 'flavours' of the operating system — System V and 4.*BSD.

However, these derivatives vary considerably in a variety of ways, both expected and unexpected. Differences exist in the behaviour of functions, their types, the type and number of arguments, location and contents of header files, etc; also the commands and utilities may differ or take different options, etc.

These differences provide headaches to authors of *portable* applications. Although it is possible to write software that will compile without modification and run correctly on a large number of existing systems, considerable expertise and knowledge of the different systems is required to do this. Acquisition of this expertise can be a time-consuming and costly overhead.

Developing software which is portable across different UNIX operating systems suffers from a major problem: the software is still likely to need modification when another new implementation of the system appears.

The POSIX System has been developed to ease this problem. The interfaces (system calls, libraries and commands) described in POSIX have evolved from those on existing UNIX systems and, where things differ on existing systems, the POSIX interfaces represent a compromise or an improvement.

The POSIX interface can be implemented on all existing UNIX systems (and, in fact on non-UNIX systems too). Porting applications to new systems will be considerably simplified, if both source and target systems are POSIX compatible.

This paper presents a technical overview of POSIX and looks at areas which differ from existing systems. The paper takes the view of an Applications Writer, but in so doing, highlights

areas which will be of interest to those responsible for making a system *POSIX-compatible*.

An overview of the position of POSIX and impact in the market place is also given.

The Andrew Toolkit — an Overview

Andrew Palay
et al.

Carnegie Mellon University

Abstract
The Andrew Toolkit is an object-orientated system designed to provide a foundation on which a large number of diverse user-interface applications can be developed. With the Toolkit, the programmer can piece together components such as text, buttons,and scroll bars to form more complex components. It also allows for the embedding of components inside other components, such as a table inside of text or a drawing inside of a table, Some of the components included in the Toolkit are multi-font text, tables, spreadsheets, drawings, equation, rasters, and simple animations. Using these components we have built a multi-media editor, a mail-system, and a help system. The Toolkit is written in C, using a simple preprocessor to provide an object-oriented environment. That environment also provides for the dynamic loading and linking of code. The dynamic facility provides a powerful extension mechanism and allows the set of components used by an application to be virtually unlimited. The Andrew Toolkit has been designed to be window-system independent. It currently runs on two window systems, including X.11, and can be ported easily to others.

Plan 9 from Bell Labs – The Network

David Leo Presotto

AT&T Bell Laboratories
Murray Hill, New jersey 07974
research!presotto
presotto@att.arpa

Abstract
This paper describes a new computing environment and the networking that underlies it. We expect the environment to accommodate either a small group or a large organisation. Although our initial implementation is targeted at 100 researchers, our goal is a system that can encompass all of AT&T's research and development.

Our design runs counter to the popular trend in computing environments, workstations connected by local area networks. We have found this solution to be both expensive and awkward. This is especially apparent in large organisations. Instead, we propose a system based on clusters of file servers and execute servers connected by high speed networks. User interfaces, similar to workstations, access the servers via lower speed distribution networks. Among other things, this simplifies administration and allows the home and work computing environment to be the same.

Formatted I/O in C++

Mark Rafter

Computer Science Department
Warwick University
Coventry
England
..!mcvax!warwick!rafter

Abstract
The *fmtio* library extends C++ stream I/O to include formatted I/O in the style of stdio. This extension is layered on top of stream I/O, and only requires minor changes to `<stream.h>`. The key traits of the original stream I/O system, namely extensibility and type-security, are retained. An example of its use is:

```
cout[ "log of %d is:%9f\n" ] << 5 << log(5);
```

which prints

```
log of 5 is: 1.609438
```

The *fmtio* library is presented as a suitable framework in which to conduct further experiments with formatted I/O systems. The methods used in the library are sketched, and its overall structure outlined. An example is given of how to equip a datatype with formatted I/O by interfacing it to the *fmtio* library.

A Protocol for the Communication between Objects

R. Schragl
UNA EDV-Beratung GmbH, München
D. Lauber
Siemens AG, München

Abstract
In object-oriented systems objects communicate with each other via messages. An object activates processing by sending a message to another object and waiting for its termination. Most of the existing implementations (e.g., SMALLTALK 80) have chosen this procedure. Normally, they are available as stand-alone systems, so that no specific protocols are required. When offering an object-oriented user interface, integrated in a conventional command-oriented system, and with tools running in a local or distributed environment, application protocols are required. This contribution defines a protocol with a service, comparable to the session-layer of the ISO reference model, suitable for this application. The characteristics of the protocol are described, and an implementation is shown within a UNIX system using the programming language C. The concepts are validated in a distributed software development environment, where system software for mainframes is developed using connected workstations based on UNIX.

UNIX V.3 and Beyond

Ian Stewartson

Data Logic Limited
System Software Development Group

Abstract
The object of this paper is to provide an overview of the current state of the UNIX Operating System environment with specific reference to the latest release (V.3) from AT&T. As UNIX has been selected as the basis for a portable operating system by a number of standards bodies, the work being done by these groups is also reviewed. Finally, the paper highlights possible and likely future developments of UNIX that are designed to improve its commercial viability.

An Overview of Miranda

David Turner

Computing Laboratory
University of Kent
Canterbury CT2 7NF
ENGLAND

Abstract
Miranda[5] is an advanced functional programming system which runs under the UNIX operating system. The aim of the Miranda system is to provide a modern functional programming language, embedded in an 'industrial quality' programming environment. It is now being used at a growing number of sites for teaching functional programming and as a vehicle for the rapid prototyping of software.

5. Miranda is a trademark of Research Software Ltd.

Erratum in Conference Proceedings

A Toolkit for Software Configuration Management

Axel Mahler
Andreas Lampen

The authors of the above paper have informed us that there were a few mistakes in the text of the paper that they supplied to be printed. Please apply these corrections in the appropriate appendices:

in Appendix A: A sample Makefile

```
#  Transformation rule definitions and
#  configuration management related rules and macros

.SUFFIXES: .h,v .c,v .h

.c,v.o:
        @-if [ -s $(SRCDIR)/$*.c ] ; \
        then \
        echo "WARNING: $*.c probably obsolete !  (RCS archive has changed)"; \
        echo "$(CC) -c $(CFLAGS) $*.c"; \
        $(CC) -c $(CFLAGS) $*.c; \
        else \
        echo temporarily checking out $*.c --- $(VID); \
        (cd $(SRCDIR); $(CO) $(COFLAGS) $*.c) > /dev/null; \
        $(CC) -c $(CFLAGS) $*.c; \
        rm $*.c; \
        fi;

.h,v.h .c,v.c:
        @-if [ -s $(SRCDIR)/$@ ] ; \
        then \
        echo "WARNING: $@ probably out of date ! (RCS archive has changed)"; \
        else \
        echo checking out $@; \
        (cd $(SRCDIR); $(CO) $(COFLAGS) $@) > /dev/null; \
        fi;

SRCDIR = /u/shape/apps
```

in Appendix B: A sample Shapefile

```
#% RULE-SECTION

fsexp:
        af*.c, attrge (state, published), attrmax (version),
                attrvar (unixfs), attrvar (debug);
        *.c, attr (state, busy), attrvar (unixfs), attrvar (debug).

fsrelease:
        *.c, attr (state, frozen), attrmax (version), attrvar (unixfs).

dbexp:
        af*.c, attrge (state, published), attrmax (version),
                attrvar (damokles), attrvar (debug);
        *.c, attr (state, busy), attrvar (damokles), attrvar (debug).

#% VARIANT-SECTION
```

Glossary

Here are the definitions of a few not-so-common English words that can be found in the newsletter. Where a word has several meanings the way in which it is used in this issue is the one explained.

abbreviation	A shortening of a word
abuse	Use in a way not intended
accolade	Praise
accrued	Increase by growth or addition
acronym	Short name made up from initial letters of long name
anticipate	For see event and act on it
awkward	Clumsy, difficult to use
celebrate	Party because of pleasant event
cement	Joing formly together
chat	Talk
citation	Writing on award
compact	Fit tightly into small space
comply	Obey rule
concept	idea of way of doing ...
concise	Complete but written in few words
convey	Carry
convince	Pususade totally in argument
culminate	Final climax of big task
customer	Alternative definition of 'Panic'
das	Variously defined - often implies movement or noise
debate	Formal spoken argument/discussion
disclaimer	Note showing that it's not your fault :-)
disclose	Tell other people about
dissipate	Waste
endeavour	Try, attempt
engrained	An assumed part of ...
epilogue	Speach/chapter at end of ...
evade	Avoid
expound	Explain at great length
extend	To make cope with ..., to make longer
freebie	Free gift
fun	Something not done seriously
funded	Paid for/by
gibberish	Something impossible to understand
glue	Sticky liquid used to join items
gossip	Talk - often unfounded information about other people
indemnify	Protect against
loaned	Given for a (short ?) period & then returned
perk	Freebie - often part of a job
pint	Unit of measure of Beer (UK)
refuse	Not accept
spotted	Seen - often with difficulty
urge	Try to persuade
utmost	Greatest possible degree
utterance	Word/saying from ...
yields	Gives/produces

AUSTRIA - UUGA
Friedrich Kofler
c/o Austro Olivetti
Rennweg 9
A-1030 Vienna
AUSTRIA

BELGIUM - BUUG
Marc Nyssen
Department of Medical Informatics
VUB, Laarbeeklaan 103
B-1090 Brussels
BELGIUM

DENMARK - DKUUG
Mogens Buhelt
Kabbelejevej 27B
DK-2700
Bronshoj
DENMARK

FINALND - FUUG
Johan Helsingius
OY Penetron Ab
Box 21
02171 ESPOO
FINLAND

FRANCE - AFUU
Miss Ann Garnery
AFUU
11 Rue Carnot
94270 Le Kremlin-Bicetre
FRANCE

GERMANY - GUUG
Mr Laengle
GUUG
Mozartstrasse 3
D-8000 Munich 2
WEST GERMANY

ICELAND - ICEUUG
Marius Olafsson
University Computer Center
Hjardarhaga 4
Reykjavik
ICELAND

IRELAND - IUUG
John Carolan
Glockenspiel Ltd
19 Belvedere Place
Dublin 1
IRELAND

ITALY - i2u
Ing Carlo Mortarino
i2u
Viale Monza 347
20126 Milano
ITALY

NETHERLANDS - NLUUG
Patricia Otter
Xirion bv
World Trade Centre
Strawinskylaan 1135
1077 XX Amsterdam
THE NETHERLANDS

NORWAY - NUUG
Jan Brandt Jensen
Unisoft A.S.
Enebakkvn 154
N-0680 Oslo 6
NORWAY

SWEDEN - EUUG-S
Hans Johansson
NCR Svenska AB
Box 4204
17104 Solna
SWEDEN

SWITZERLAND - UNIGS
Professor Wolfgang Fitchner
Institute for Integrated Systems
ETH Zentrum
CH-8092 Zurich
SWITZERLAND

UNITED KINGDOM - UKUUG
Bill Barrett
Owles Hall
Buntingford
Hertfordshire SG9 9PL
UNITED KINGDOM

The European UNIX systems User Group
Owles Hall
Buntingford
Hertfordshire SG9 9PL
UK
+44 763 73039