NEWSLETTER

EUUG

European UNIX® systems User Group

Volume 10, No. 3
Autumn 1990

CONTENTS

EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

EUUG

Volume 10, Number 3
Autumn 1990

Copyright © EUUG 1990.

This document is copyright of the EUUG. In addition, certain information contained herein may be covered by one or more licence or non-disclosure agreements. Copying of individual articles for strictly personal use or for non commercial use within an organisation or institution which is a current member of the EUUG is permitted. All other copying is prohibited without the prior permission in writing from the EUUG.

Abstracting with credits is permitted.

The C++ Column is joint copyright with Mr Mark Rafter.

ISSN 1011-4211

Editorial Team:

Frances Brazier Publications Executive	Department of Cognitive Psychology Vrije Universiteit de Boelelaan 1115, 1081 HV Amsterdam. The Netherlands frances@psy.vu.nl
Alain Williams Editor	Parliament Hill Computers Ltd. 7 Prospect Street Caversham Berkshire RG4 8JB U.K. addw@phcomp.co.uk Tel: +44 734 461232 Fax: +44 734 474194

Typesetting:

Gina Baikenycz	The Instruction Set Ltd. City House 190 City Road London EC1V 2QH U.K. gina@inset.co.uk

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@EU.net Tel: +44 763 73039 Fax: +44 763 73255

UNIX is a registered trademark of UNIX system laboratories in the USA and other countries
X Window System is a trademark of MIT

Editorial

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

Parliament Hill Computers Ltd

Son of UNIX

Do you remember the days when UNIX was a nice small system that ran well on PDP-11's? Are you aghast at the way that UNIX has grown big?

Plan 9 is the new operating system that is just emerging from Bell Labs. It is nice and small.

Rob Pike describes Plan 9 on page 2. He is wants it to escape the stagnating clutches of the standards committees and allow it to grow.

Tom Duff describes a new command interpretter for Plan 9 on page 12.

These are two of the Plan 9 papers that were delivered at the recent UKUUG Summer Conference and gave us much to talk about. You will find the abstracts for the conference on page 103.

The most succinct evaluation that I remember was "I want it".

EUUG Conferences

The next EUUG conference is in October in Nice, France. You will find details and a booking form on page 23.

Looking ahead to next May you will find the call for papers for the conference in Tromsø, Norway on page. Why not submit a paper – if it is accepted your boss will *have* to let you go to present it.

If you want to know how good the last EUUG conference – read Guenther Fischer's view on the conference on page 35, with the abstracts on page 95.

C++ Column

Please welcome Mark Rafter who is restarting the C++ column on page 71.

This issue he reviews what looks like *the* C++ reference manual.

"Thank You" Sun

The EUUG would like to thank SUN Microsystems Nederland for the XI Logics Disk Controller which they have just donated to the EUUG as an extension to the *mcsun* configuration donated previously.

This will help with the continued growth and expansion of services offered to the users of EUnet.

Advertising

There are 5,500 copies if this newsletter printed, and it has a Europe wide circulation.

It will cost you £300 per full page plus £100 for every extra colour that you use. The copy dates are the same as for editorial content.

Items may also be inserted into the envelope with the newsletter. This will cost you £300 for an A4 sheet, larger amounts will depend on the weight of the insert.

Make people thougout Europe aware of your company and its products.

Contact myself or Owles Hall for more information.

Free Advertising

When typesetting this newsletter we quite often have a half page left over at the end of an article.

I am quite happy to accept small adverts to fill these up. There is no charge for it – but no guarantee that it will be printed.

Contact me to discuss this.

Future EUUGN Dates

The publication dates and the copy dates for future issues of the EUUGN are printed below:

	Copy Date	Publication Date
	15 October	1 December
1991	14 January	1 March
	15 April	1 June

Plan 9 from Bell Labs

Rob Pike, Dave Presotto, Ken Thompson, Howard Trickey
rob@research.att.com

Bell Labs

Plan 9 is a distributed computing environment. It is assembled from separate machines acting as CPU servers, file servers, and terminals. The pieces are connected by a single file-oriented protocol and local name space operations. By building the system from distinct, specialised components rather than from similar general-purpose components, Plan 9 achieves levels of efficiency, security, simplicity, and reliability seldom realised in other distributed systems. This paper discusses the building blocks, interconnections, and conventions of Plan 9.

Introduction[1]

Plan 9 is a general-purpose, multi-user, portable distributed system implemented on a variety of computers and networks. It lacks a number of features often found in other distributed systems, including

i. A uniform distributed name space,

ii. Process migration,

iii. Lightweight processes,

iv. Distributed file caching,

v. Personalised workstations,

vi. Support for X windows.

Unhappy with the trends in commercial systems, we began a few years ago to design a system that could adapt well to changes in computing hardware. In particular, we wanted to build a system that could profit from continuing improvements in personal machines with bitmap graphics, in medium- and high-speed networks, and in high-performance microprocessors. A common approach is to connect a group of small personal timesharing systems – workstations – by a medium-speed network, but this has a number of failings. Because each workstation has private data, each must be administered separately; maintenance is difficult to centralise. The machines are replaced every couple of years to take advantage of technological improvements, rendering the hardware obsolete often before it has been paid for. Most telling, a workstation is a largely self-contained system, not specialised to any particular task, too slow and I/O-bound for fast compilation, too expensive to be used just to run a window system. For our purposes, primarily software development, it seemed that an approach based on distributed specialisation rather than compromise could better address issues of cost-effectiveness, maintenance, performance, reliability, and security. We decided to build a completely new system, including compiler, operating system, networking software, command interpreter, window system, and (on the hardware side) terminal. This construction would also offer an occasion to rethink, revisit, and perhaps even replace most of the utilities we had accumulated over the years.

1. This paper is reprinted with kind permission of the UKUUG and was delivered at the UKUUG conference in London in July 1990. See the end of this newsletter for abstracts of this conference.

Plan 9 is divided along lines of service function. CPU servers concentrate computing power into large (not overloaded) multiprocessors; file servers provide repositories for storage; and terminals give each user of the system a dedicated computer with bitmap screen and mouse on which to run a window system. The sharing of computing and file storage services provides a sense of community for a group of programmers, amortises costs, and centralises and hence simplifies management and administration.

The pieces communicate by a single protocol, built above a reliable data transport layer offered by an appropriate network, that defines each service as a rooted tree of files. Even for services not usually considered as files, the unified design permits some noteworthy and profitable simplification. Each process has a local file *name space* that contains attachments to all services the process is using and thereby to the files in those services. One of the most important jobs of a terminal is to support its user's customised view of the entire system as represented by the services visible in the name space.

To be used effectively, the system requires a CPU server, a file server, and a terminal; it is intended to provide service at the level of a departmental computer centre or larger. The CPU server and file server are large machines best housed in an air conditioned machine room with conditioned power. The system's strengths stem in part from economies of scale, and the scale we have in mind is large. One of our goals, perhaps unrealisable, is to unite the computing environment for all of AT&T Bell Laboratories (about 30,000 people) into a single Plan 9 system comprising thousands of CPU and file servers spread throughout, and clustered in, the company's various departments. That is clearly beyond the administrative capacity of workstations on Ethernets.

The following sections describe the basic components of Plan 9, explain the name space and how it is used, and offer some examples of unusual services that illustrate how the ideas of Plan 9 can be applied to a variety of problems.

CPU Servers

Several computers provide CPU service for Plan 9. The production CPU server is a Silicon Graphics Power Series machine with four 25MHz MIPS processors, 128 megabytes of memory, no disk, and a 20 megabyte-per-second back-to-back DMA connection to the file server. It also has Datakit and Ethernet controllers to connect to terminals and non-Plan 9 systems [Fra80a, Met80a]. The operating system provides a conventional view of processes, based on `fork` and `exec` system calls [Ker84a], and of files, mostly determined by the remote file server. Once a connection to the CPU server is established, the user may begin typing commands to a command interpreter in a conventional looking environment [Duf90a, Rit74a].

A multiprocessor CPU server has several advantages. The most important is its ability to absorb load. If the machine is not saturated (which can be economically feasible for a multiprocessor) there is usually a free processor ready to run a new process. This is similar to the notion of free disk blocks in which to store new files on a file system. The comparison extends farther: just as one might buy a new disk when a file system gets full, one may add processors to a multiprocessor when the system gets busy, without needing to replace or duplicate the entire system. Of course, one may also add new CPU servers and share the file servers.

The CPU server performs compilation, text processing, and other applications. It has no local storage; all the permanent files it accesses are provided by remote servers. Transient parts of the name space, such as the collected images of active processes [Kil84a] or services provided by user processes, may reside locally but these disappear when the CPU server is rebooted. Plan 9 CPU servers are as interchangeable for their task – computation – as are ordinary terminals for theirs.

File Servers

The Plan 9 file servers hold all permanent files. The current server is another Silicon Graphics computer with two processors, 64 megabytes of memory, 600 megabytes of magnetic disk, and a 300 gigabyte jukebox of write-once optical disk (WORM). (This machine is to be replaced by a MIPS 6280, a single processor with much greater I/O bandwidth.) It connects to Plan 9 CPU servers through 20 megabyte-per-second DMA links, and to terminals and other machines through conventional networks.

The file server presents to its clients a file system rather than, say, an array of disks or blocks or files. The files are named by slash-separated components that label branches of a tree, and may be addressed for I/O at the byte level. The location of a file in the server is invisible to the client. The true file system resides on the WORM, and is accessed through a two-level cache of magnetic disk and RAM. The contents of recently-used files reside in RAM and are sent to the CPU server rapidly by DMA over a high-speed link, which is much faster than regular disk although not as fast as local memory. The magnetic disk acts as a cache for the WORM and simultaneously as a backup medium for the RAM. With the high-speed links, it is unnecessary for clients to cache data; instead the file server centralises the caching for all its clients, avoiding the problems of distributed caches.

The file server actually presents several file systems. One, the "main" system, is used as the file system for most clients. Other systems provide less generally-used data for private applications. One service is unusual: the backup system. Once a day, the file server freezes activity on the main file system and flushes the data in that system to the WORM. Normal file service continues unaffected, but changes to files are applied to a fresh hierarchy, fabricated on demand, using a copy-on-write scheme [Qui90a]. Thus, the file tree is split into two: a read-only version representing the system at the time of the dump, and an ordinary system that continues to provide normal service. The roots of these old file trees are available as directories in a file system that may be accessed exactly as any other (read-only) system. For example, the file `/usr/rob/doc/plan9.ms` as it existed on April 1, 1990, can be accessed through the backup file system by the name `/1990/0401/usr/rob/doc/plan9.ms`. This scheme permits recovery or comparison of lost files by traditional commands such as file copy and comparison routines rather than by special utilities in a backup subsystem. Moreover, the backup system is provided by the same file server and the same mechanism as the original files so permissions in the backup system are identical to those in the main system; one cannot use the backup data to subvert security.

Terminals

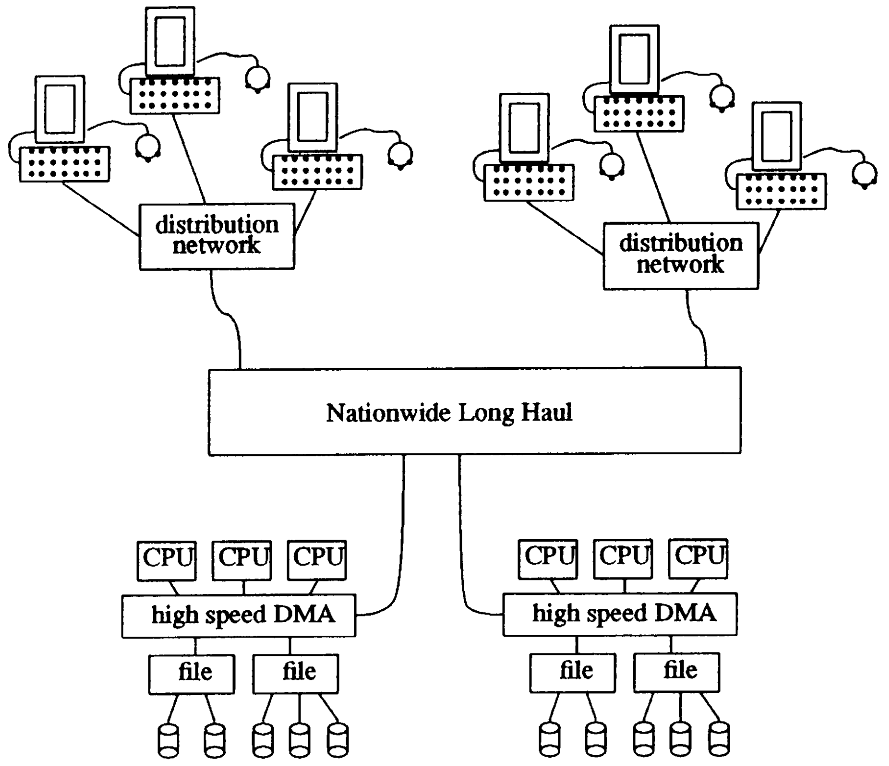
The standard terminal for Plan 9 is a Gnot (with silent "G"), a locally-designed machine of which several hundred have been manufactured. The terminal's hardware is reminiscent of a diskless workstation: 4 or 8 megabytes of memory, a 25MHz 68020 processor, a 1024×1024 pixel display with two bits per pixel, a keyboard, and a mouse. It has no external storage and no expansion bus; it is a terminal, not a workstation. A 2 megabit per second packet-switched distribution network connects the terminals to the CPU and file servers. Although the bandwidth is low for applications such as compilation, it is more than adequate for the terminal's intended purpose: to provide a window system, that is, a multiplexed interface to the rest of Plan 9.

Unlike a workstation, the Gnot does not handle compilation; that is done by the CPU server. The terminal runs a version of the CPU server's operating system, configured for a single, smaller processor with support for bitmap graphics, and uses that to run programs such as a window system and a text editor. Files are provided by the standard file server over the terminal's network connection.

Just like old character terminals, all Gnots are equivalent, as they have no private storage either locally or on the file server. They are inexpensive enough that every member of our research centre can have two: one at work and one at home. A person working on a Gnot at home sees exactly the same system as at work, as all the files and computing resources remain at work where they can be shared and maintained effectively.

Networks

Plan 9 has a variety of networks that connect the components. CPU servers and file servers communicate over back-to-back DMA controllers. That is only practical for the scale of, say, a computer centre or departmental computing resource. More distant machines are connected by traditional networks such as Ethernet or Datakit. A terminal or CPU server may use a remote file server completely transparently except for performance considerations. As our Datakit network spans the country, Plan 9 systems could be assembled on a large scale, although this has not been tried in practice. (See Figure 1.)



To keep their cost down, Gnots employ an inexpensive network that uses standard telephone wire and a single-chip interface. (The throughput is respectable, about 120 kilobytes per second.)

To get even that bandwidth to home is of course problematic. Some of us have DS-1 lines at 1.54 megabits per second; others are experimenting with more modest communications equipment. Since the terminal only mediates communication – it instructs the CPU server to connect to the file server but does not participate in the resulting communication – the relatively low bandwidth to the terminal does not affect the overall performance of the system.

Name Spaces

There are two kinds of name space in Plan 9: the global space of the names of the various servers on the network and the local space of files and servers visible to a process. Names of machines and services connected to Datakit are hierarchical, for example `nj/mh/astro/helix`, defining (roughly) the area, building, department, and machine in a department [Fra80a]. Because the network provides naming for its machines, global naming issues need not be handled directly by Plan 9. However one of Plan 9's fundamental operations is to attach network services to the local name space on a per-process basis. This fine-grained control of the local name space is used to address issues of customisability, transparency, and heterogeneity.

The protocol for communicating with Plan 9 services is file-oriented; all services must implement a file system. That is, each service, local or remote, is arranged into a set of file-like objects collected into a hierarchy called the name space of the server. For a file server, this is a trivial requirement. Other services must sometimes be more imaginative. For instance, a printing service might be implemented as a directory in which processes create files to be printed. Other examples are described in the following sections; for the moment, consider just a set of ordinary file servers distributed around the network.

When a program calls a Plan 9 service (using mechanisms inherent in the network and outside Plan 9 itself) the program is connected to the root of the name space of the service. Using the protocol, usually as mediated by the local operating system into a set of file-oriented system calls, the program accesses the service by opening, creating, removing, reading, and writing files in the name space.

From the set of services available on the network, a user of Plan 9 selects those desired: a file server where personal files reside, perhaps other file servers where data is kept, or a departmental file server where the software for a group project is being written. The name spaces of these various services are collected and joined to the user's own private name space by a fundamental Plan 9 operator, called *attach*, that joins a service's name space to a user's. The user's name space is formed by the union of the spaces of the services being used. The local name space is assembled by the local operating system for each user, typically by the terminal. The name space is modifiable on a per-process level, although in practice the name space is assembled at log-in time and shared by all that user's processes.

To log in to the system, a user sits at a terminal and instructs it which file server to connect to. The terminal calls the server, authenticates the user (see below), and loads the operating system from the server. It then reads a file, called the *profile*, in the user's personal directory. The profile contains commands that define what services are to be used by default and where in the local name space they are to be attached. For example, the main file server to be used is attached to the root of the local name space, `/`, and the process file system is attached to the directory `/proc`. The profile then typically starts the window system.

Within each window in the window system runs a command interpreter that may be used to execute commands locally, using file names interpreted in the name space assembled by the profile. For computation-intensive applications such as compilation, the user runs a command `cpu` that selects (automatically or by name) a CPU server to run commands. After typing `cpu`, the user sees a regular prompt from the command interpreter. But that command interpreter is running on the CPU server *in the same name space – even the same current directory – as the* `cpu` *command itself*. The terminal exports a description of the name space to the CPU server, which then assembles an identical name space, so the customised view of the system assembled by the terminal is the same as that seen on the CPU server. (A description of the name space is used rather than the name space itself so the CPU server may use high-speed links when possible rather than requiring intervention by the terminal.) The `cpu` command affects only the performance of subsequent commands; it has nothing to do with the services available or how they are accessed.

Although there is a large catalogue of services available in Plan 9, including the service that finds services, a few suffice to illustrate the usage and possibilities of this design.

The Process File System

An example of a local service is the "process file system", which permits examination and debugging of executing processes through a file-oriented interface. It is related to Killian's process file system [Kil84a] but its differences exemplify the way that Plan 9 services are constructed.

The root of the process file system is conventionally attached to the directory `/proc`. (Convention is important in Plan 9; although the name space may be assembled willy-nilly, many programs have conventional names built in that require the name space to have a certain form. It doesn't matter which server the program `/bin/rc` (the command interpreter) comes from but it must have that name to be accessible by the commands that call on it.) After attachment, the directory `/proc` itself contains one subdirectory for each local process in the system, with name equal to the numerical unique identifier of that process. (Processes running on the remote CPU server may also be made visible; this will be discussed below.) Each subdirectory contains a set of files that implement the view of that process. For example, `/proc/77/mem` contains an image of the virtual memory of process number 77. That file is closely related to the files in Killian's process file system, but unlike Killian's, Plan 9's `/proc` implements other functions through other files rather than through peculiar operations applied to a single file. Here is a list of the files provided for each process.

`mem`
: The virtual memory of the process image. Offsets in the file correspond to virtual addresses in the process.

`ctl`
: Control behaviour of the processes. Messages sent (by a `write` system call) to this file

cause the process to stop, terminate, resume execution, etc.

`text`
: The file from which the program originated. This is typically used by a debugger to examine the symbol table of the target process, but is in all respects except name the original file; thus one may type "`/proc/77/text`" to the command interpreter to instantiate the program afresh.

`note`
: Any process with suitable permissions may write the `note` file of another process to send it an asynchronous message for interprocess communication. The system also uses this file to send (poisoned) messages when a process misbehaves, for example divides by zero.

`status`
: A fixed-format ASCII representation of the status of the process. It includes the name of the file the process was executed from, the CPU time it has consumed, its current state, etc.

The `status` file illustrates how heterogeneity and portability can be handled by a file server model for system functions. The command `cat /proc/*/status` presents (readably but somewhat clumsily) the status of all processes in the system; in fact the process status command `ps` is just a reformatting of the ASCII text so gathered. The source for `ps` is a page long and is completely portable across machines. Even when `/proc` contains files for processes on several heterogeneous machines, the same implementation works.

Whether the functions provided by the `ctl` file should instead be accessed through further files – `stop`, `terminate`, etc. – is a matter of taste. We chose to fold all the true control operations into the `ctl` file and provide the more data-intensive functions through separate files.

It is worth noting that the services `/proc` provides, although varied, do not strain the notion of a process as a file. For example, it is not possible to terminate a process by attempting to remove its process file nor is it possible to start a new process by creating a process file. The files give an active view of the processes, but they do not literally represent them. This distinction is important when designing services as file systems.

The Window System

In Plan 9, user programs, as well as specialised stand-alone servers, may provide file service. The window system is an example of such a program; one of Plan 9's most unusual aspects is that the window system is implemented as a user-level file server.

The window system is a server that presents a file `/dev/cons`, similar to the `/dev/tty` or `CON:` of other systems, to the client processes running in its windows. Because it controls all I/O activities on that file, it can arrange that each window's group of processes sees a private `/dev/cons`. When a new window is made, the window system allocates a new `/dev/cons` file, puts it in a new name space (otherwise the same as its own) for the new client, and begins a client process in that window. That process connects the standard input and output channels to `/dev/cons` using the normal file opening system call and executes a command interpreter. When the command interpreter prints a prompt, it will therefore be written to `/dev/cons` and appear in the appropriate window.

It is instructive to compare this structure to other operating systems. Most operating systems provide a file like `/dev/cons` that is an alias for the terminal connected to a process. A process that opens the special file accesses the terminal it is running on without knowing the terminal's precise name. Since the alias is usually provided by special arrangement in the operating system, it can be difficult for a window system to guarantee that its client processes can access their window through this file. This issue is handled easily in Plan 9 by inverting the problem. A set of processes in a window shares a name space and in particular `/dev/cons`, so by multiplexing `/dev/cons` and forcing all textual input and output to go through that file the window system can simulate the expected properties of the file.

The window system serves several files, all conventionally attached to the directory of I/O devices, `/dev`. These include `cons`, the port for ASCII I/O; `mouse`, a file that reports the position of the mouse; and `bitblt`, which may be written messages to execute bitmap graphics primitives. Much as the different `cons` files keep separate clients' output in separate windows, the `mouse` and `bitblt` files are implemented by the window system in a way that keeps the various clients independent. For example, when a client process in a window writes a message (to the `bitblt` file) to clear the screen, the window system clears only that window. All graphics sent to partially or totally obscured windows is maintained as a bitmap layer, in memory private to the window system [Pik83a]. The clients are oblivious of one another.

Since the window system is implemented entirely at user level with file and name space operations, it can be run recursively: it may be a client of itself. The window system functions by opening the files `/dev/cons`, `/dev/bitblt`, etc., as provided by the operating system, and reproduces – multiplexes – their functionality among its clients. Therefore, if a fresh instantiation of the window system is run in a window, it will behave normally, multiplexing *its* `/dev/cons` and other files for *its* clients. This recursion can be used profitably to debug a new window system in a window or to multiplex the connection to a CPU server [Pik89a]. Since the window system has no bitmap graphics code – all its graphics operations are executed by writing standard messages to a file – the window system may be run on any machine that has `/dev/bitblt` in its name space, including the CPU server.

CPU Command

The `cpu` command connects from a terminal to a CPU server using a full-duplex network connection and runs a setup process there. The terminal and CPU processes exchange information about the user and name space, and then the terminal-resident process becomes a user-level file server that makes the terminal's private files visible from the CPU server. (At the time of writing, the CPU server builds the name space by re-executing the user's profile; a version being designed will export the name space using a special terminal-resident server that can be queried to recover the terminal's name space.) The CPU process makes a few adjustments to the name space, such as making the file `/dev/cons` on the CPU server *be the same file as on the terminal*, perhaps making both the local and remote process file system visible in `/proc`, and begins a command interpreter. The command interpreter then reads commands from, and prints results on, its file `/dev/cons`, which is connected through the terminal process to the appropriate window (for example) on the terminal. Graphics programs such as bitmap editors also may be executed on the CPU server since their definition is entirely based on I/O to files "served" by the terminal for the CPU server. The connection to the CPU server and back again is utterly transparent.

This connection raises the issue of heterogeneity: the CPU server and the terminal may be, and in the current system are, different types of processors. There are two distinct problems: binary data and executable code. Binary data can be handled two ways: by making it not binary or by strictly defining the format of the data at the byte level. The former is exemplified by the `status` file in `/proc`, which enables programs to examine, transparently and portably, the status of remote processes. Another example is the file, provided by the terminal's operating system, `/dev/time`. This is a fixed-format ASCII representation of the number of seconds since the epoch that serves as a time base for `make` and other programs [Ker84a]. Processes on the CPU server get their time base from the terminal, thereby obviating problems of distributed clocks.

For files that are I/O intensive, such as `/dev/bitblt`, the overhead of an ASCII interface can be prohibitive. In Plan 9, such files therefore accept a binary format in which the byte order is predefined, and programs that access the files use portable libraries that make no assumptions about the order. Thus `/dev/bitblt` is usable from any machine, not just the terminal. This principle is used throughout Plan 9. For instance, the format of the compilers' object files and libraries is similarly defined, which means that object files are independent of the type of the CPU that compiled them.

Having different formats of executable binaries is a thornier problem, and Plan 9 solves it adequately if not gracefully. Directories of executable binaries are named appropriately: `/mips/bin`, `/68020/bin`, etc., and a program may ascertain, through a special server, what CPU type it is running on. A program, in particular the `cpu` command, may therefore attach the appropriate directory to the conventional name `/bin` so that when a program runs, say, `/bin/rc`, the appropriate file is found. Although this is a fairly clumsy solution, it works well in practice. The various object files and compilers use distinct formats and naming conventions, which makes cross-compilation painless, at least once automated by `make` or a similar program

[Ker84a].

Security

Plan 9 does not address security issues directly, but some of its aspects are relevant to the topic. Breaking the file server away from the CPU server enhances the possibilities for security. As the file server is a separate machine that can only be accessed over the network by the standard protocol, and therefore can only serve files, it cannot run programs. Many security issues are resolved by the simple observation that the CPU server and file server communicate using a rigorously controlled interface through which it is impossible to gain special privileges.

Of course, certain administrative functions must be performed on the file server, but these are available only through a special command interface accessible only on the console and hence subject to physical security. Moreover, that interface is for administration only. For example, it permits making backups and creating and removing files, but it does not permit reading files or changing their permissions. *The contents of a file with read permission for only its owner will not be divulged by the file server to any other user, even the administrator.*

Of course, this begs the question of how a user proves who he or she is. At the moment, we use a simple authentication manager on the Datakit network itself, so that when a user logs in from a terminal, the network assures the authenticity of the maker of calls from the associated terminal. In order to remove the need for trust in our local network, we plan to replace the authentication manager by a Kerberos-like system [Mil87a].

Discussion

A fairly complete version of Plan 9 was built in 1987 and 1988, and then its development was abandoned for a number of aesthetic and technical reasons. In May of 1989 work was begun on a completely new system, based on the SGI MIPS-based multiprocessors, using the first version as a bootstrap environment. By October, the CPU server could compile all its own software, using the first-draft file server. The SGI file server came on line in February 1990; the true operating system kernel at its core was taken from the CPU server's system, but the file server is otherwise a completely separate program (and computer). The CPU server's system was ported to the 68020 in 13 hours elapsed time on November 12-13, 1989. One portability bug was found; the fix affected two lines of code. At the time of writing (April 1990), work has just begun on the new window system; it should be running well before this paper appears (July 1990). (Until it is complete, we will continue to use the terminal software from the 1987-1988 implementation.)

All the authors use Plan 9 almost exclusively; only the lack of an electronic mail facility, which is being addressed, prevents us from moving over permanently. Plan 9 is up and running and comfortable to use, although it is certainly too early to pass final judgement.

The multiprocessor operating system for the MIPS-based CPU server has 454 lines of assembly language, more than half of which saves and restores registers on interrupts. The kernel proper contains 3647 lines of C plus 774 lines of header files, which includes all process control, virtual memory support, trap handling, and so on. There are 1020 lines of code to interface to the 29 system calls. Much of the functionality of the system is contained in the "drivers" that implement built-in servers such as `/proc`; these and the network software add another 9511 lines of code. Most of this code is identical on the 68020 version; for instance, all the code to implement processes, including the process switcher and the `fork` and `exec` system calls, is identical in the two versions; the peculiar properties of each processor are encapsulated in two five-line assembler routines. (The code for the respective MMU's is quite different, although the page fault handler is substantially the same.) It is only fair to admit, however, that the compilers for the two machines are closely related, and the operating system may depend on properties of the compiler in unknown ways.

The system is efficient. On the four-processor machine connected to the MIPS file server, the 45 source files of the operating system compile in about ten seconds of real time and load in another ten. (The loader runs single-threaded.) Partly due to the register-saving convention of the compiler, the null system call takes only 7 microseconds on the MIPS, about half of which is attributed to relatively slow memory on the multiprocessor. A process fork takes 700 microseconds irrespective of the process's size.

Plan 9 does not implement lightweight processes explicitly. We are uneasy about deciding where on the continuum from fine-grained hardware-supported parallelism to the usual timesharing notion of a process we should provide support for user multiprocessing. Existing definitions of threads and lightweight processes seem arbitrary and raise more questions than they resolve [Acc86a]. We prefer to have a single kind of process and to permit multiple processes to share their address space. With the ability to share local memory and with efficient process creation and switching, both of which are in Plan 9, we can match the functionality of threads without taking a stand on how users should multiprocess.

Process migration is also deliberately absent from Plan 9. Although Plan 9 makes it easy to instantiate processes where they can most effectively run, it does nothing explicit to make this happen. The compiler, for instance, does not arrange that it run on the CPU server. We prefer to do coarse-grained allocation of computing resources simply by running each new command interpreter on a lightly-loaded CPU server. Reasonable management of computing resources renders process migration unnecessary.

Other aspects of the system lead to other efficiencies. A large single-threaded chess database problem runs about four times as fast on Plan 9 as on the same machine running commercial software because the remote cache on the file server is so large. In general, most file I/O is done by direct DMA from the file server's cache; the file server rarely needs to read from disk at all.

Much of Plan 9 is straightforward. The individual pieces that make it up are relatively ordinary; its unusual aspects are in how the pieces are put together. As a case in point, the recent interest in using X terminals connected to timeshared hosts might seem to be similar in spirit to how Plan 9 terminals are used, but that is a mistaken impression. The Gnot, although similar in hardware power to a typical X terminal, serves a much higher-level function in the computing environment. It is a fully programmable computer running a virtual memory operating system that maintains its user's view of the entire Plan 9 system. It offloads from the CPU server all the bookkeeping and I/O intensive chores that a window system must perform. It is not really a workstation either; for example one would rarely bother to compile on the Gnot, although one would certainly run a text editor there. Like the other pieces of Plan 9, the Gnot's strength derives from careful specialisation in concert with other specialised components.

Acknowledgements

Many people helped build the system. We would like especially to thank Bart Locanthi, who built the Gnot and encouraged us to program it; Tom Duff, who wrote the command interpreter `rc`, Tom Killian and Ted Kowalski, who cheerfully endured early versions of the software; and Dennis Ritchie, who frequently provided us with much-needed wisdom.

Authors' Note

Since this paper was first written almost all the 'to be done' work mentioned in the paper, in particular the window system, has been done.

References

[Acc86a] M. J. Accetta, Robert Baron, William Bolosky, David Golub, Richard Rashid, Avadis Tevanian, and Michael Young, "Mach: A New Kernel Foundation for UNIX Development" , *USENIX Conference Proceedings*, Atlanta, GA (July, 1986).

[Duf90a] A T. Duff, "Rc – A Shell for Plan 9 and UNIX", *UNIX Programmer's Manual, Tenth Edition*, Murray Hill, NJ, AT&T Bell Laboratories, (1990).

[Fra80a] A. G. Fraser, "Datakit – A Modular Network for Synchronous and Asynchronous Traffic", *Proc. Int. Conf. on Commun.*, Boston, MA, (June 1980).

[Ker84a] Brian W. Kernighan, and Rob Pike, *The UNIX Programming Environment*, Prentice-Hall, Englewood Cliffs, NJ, (1984).

[Kil84a] T. J. Killian, "Processes as Files", *USENIX Summer Conference Proceedings*, Salt Lake City, UT, USA, (June 1984).

[Met80a] R. M Metcalfe, and D. R. Boggs, *The Ethernet Local Network: Three Reports*, XEROX Palo Alto Research Center, (February 1980).

[Mil87a] A. S. P. Miller, C. Neumann, J. I. Schiller, and J. H. Saltzer, *Kerberos Authentication and Authorization System*, MIT, (1987).

[Pik83a] R. Pike, ''Graphics in Overlapping Bitmap Layers'', *Transactions on Graphics* 2(2), pp. 135-160 (1983).

[Pik89a] R. Pike, ''A Concurrent Window System'', *Computing Systems* 2(2), pp. 133-153 (1989).

[Qui90a] S. Quinlan, ''A Cached WORM File System'', *Software – Practice and Experience*, p. To appear (1990).

[Rit74a] D. M. Ritchie, and K. Thompson, ''The UNIX Time-Sharing System'', *Comm. Assoc. Comp. Mach.* 17(7), (July 1974).

EXUG

The European X User Group

X WINDOW SYSTEM CONFERENCE

24-26 September 1990
The University of Surrey, Guildford, England.

Tutorials: There will be three Tutorials ranging from Introductory to Advanced
Paul Asente (author of the X Toolkit) will lead the tutorial on tookit programming.

Speakers include several of the leading international figures in X, some of the original developers of X, and users who will present case-studies of their work.

Costs to EXUG members :	£250 residential
	£220 non-residential
Cost to non-members:	£300 residential
	£280 non-residential
Exhibition stands (available only to corporate members) :	£500

There are some en suite rooms available at a premium of £15 per night booked.
A Delegate pack including car parking permits will be sent upon receipt of fees.
All bills incur a VAT charge of 15%.

Bookings to: The Secretary, EXUG, 185 High Street, Cottenham, Cambridge CB4 4RX. ENGLAND.

Rc – A Shell for Plan 9 and UNIX Systems

Tom Duff
td@research.att.com

AT&T Bell Laboratories

Rc is a command interpreter for Plan 9. It also runs on a variety of traditional systems, including SunOS and the Tenth Edition. It provides similar facilities to Bourne's */bin/sh*, with some small additions and mostly less idiosyncratic syntax. This paper introduces *rc*'s highlights with numerous examples, and discusses its design and why it varies from Bourne's.

Introduction[1]

Plan 9 needs a command-programming language. As porting the Bourne shell to an incompatible new environment seemed a daunting task, I chose to write a new command interpreter, called *rc* because it runs commands. Although tinkering with perfection is a dangerous business, I could hardly resist trying to "improve" on Bourne's design. Thus *rc* is similar in spirit but different in detail from Bourne's shell.

The bulk of this paper describes *rc*'s principal features with many small examples and a few larger ones. We close with a discussion of the principles guiding *rc*'s design and why it differs from Bourne's design. The descriptive sections include little discussion of the rationale for particular features, as individual details are hard to justify in isolation. The impatient reader may wish to skip to the discussion at the end before skimming the expository parts of the paper.

Simple commands

For the simplest uses *rc* has syntax familiar to Bourne-shell users. Thus all of the following behave as expected:

```
date
con alice
who >user.names
who >>user.names
wc <file
echo [a-f]*.c
who | wc
who; date
cc *.c &
cyntax *.c && cc -g -o cmd *.c
rm -r junk || echo rm failed!
```

Quotation

An argument that contains a space or one of *rc*'s other syntax characters must be enclosed in apostrophes (`'`):

```
rm 'odd file name'
```

An apostrophe in a quoted argument must be doubled:

```
echo 'How''s your father?'
```

Variables

Rc provides variables whose values are lists of arguments. Variables may be given values by typing, for example:

```
path=(. /bin /usr/bin)
user=td
tty=/dev/tty8
```

The parentheses indicate that the value assigned to `path` is a list of three strings. The variables

1. This paper is reprinted with kind permission of the UKUUG and was delivered at the UKUUG conference in London in July 1990. See the end of this newsletter for abstracts of this conference.

`user` and `tty` are assigned lists containing a single string.

The value of a variable can be substituted into a command by preceding its name with a `$`, like this:

```
echo $path
```

If `path` had been set as above, this would be equivalent to

```
echo . /bin /usr/bin
```

Variables may be subscripted by numbers or lists of numbers, like this:

```
echo $path(2)
echo $path(3 2 1)
```

These are equivalent to

```
echo /bin
echo /usr/bin /bin .
```

There can be no space separating the variable's name from the left parenthesis. Otherwise, the subscript would be considered a separate parenthesized list.

The number of strings in a variable can be determined by the `$#` operator. For example,

```
echo $#path
```

would print the number of entries in `$path`.

The following two assignments are subtly different:

```
empty=()
null=''
```

The first sets `empty` to a list containing no strings. The second sets `null` to a list containing a single string, but the string contains no characters.

Although these may seem like more or less the same thing (in Bourne's shell, they are indistinguishable), they behave differently in almost all circumstances. Among other things

```
echo $#empty
```

prints 0, whereas

```
echo $#null
```

prints 1.

All variables that have never been set have the value `()`.

Arguments

When *rc* is reading its input from a file, the file has access to the arguments supplied on *rc*'s command line. The variable `$*` initially has the list of arguments assigned to it. The names `$1`, `$2`, etc. are synonyms for `$*(1)`, `$*(2)`, etc. In addition, `$0` is the name of the file from which *rc*'s input is being read.

Concatenation

Rc has a string concatenation operator, the caret `^`, to build arguments out of pieces.

```
echo hully^gully
```

is exactly equivalent to

```
echo hullygully
```

Suppose variable `i` contains the name of a command. Then

```
cc -o $i $i^.c
```

might compile the command's source code, leaving the result in the appropriate file.

Concatenation distributes over lists. The following

```
echo (a b c)^(1 2 3)
src=(main subr io)
cc $src^.c
```

are equivalent to

```
echo a1 b2 c3
cc main.c subr.c io.c
```

In detail, the rule is: if both operands of `^` are lists of the same non-zero number of strings, they are concatenated pairwise. Otherwise, if one of the operands is a single string, it is concatenated with each member of the other operand in turn. Any other combination of operands is an error.

Free carets

User demand has dictated that *rc* insert carets in certain places, to make the syntax look more like the Bourne shell. For example, this:

```
cc -$flags $stems.c
```

is equivalent to

```
cc -^$flags $stems^.c
```

In general, *rc* will insert `^` between two arguments that are not separated by white space. Specifically, whenever one of `$'`` follows a quoted or unquoted word, or an unquoted word follows a quoted word with no intervening blanks or tabs, a `^` is inserted between the two. If an unquoted word immediately following a `$` contains a character other than an alphanumeric, underscore or `*`, a `^` is inserted before the first such character.

Command substitution

It is often useful to build an argument list from the output of a command. *Rc* allows a command, enclosed in braces and preceded by a left quote, `` `{...} ``, anywhere that an argument is required. The command is executed and its standard output captured. The characters stored in the variable `ifs` are used to split the output into arguments. For example,

```
cat `{ls -tr|sed 10q}
```

will catenate the ten oldest files in the current directory in temporal order.

Pipeline branching

The normal pipeline notation is general enough for almost all cases. Very occasionally it is useful to have pipelines that are not linear. Pipeline topologies more general than trees can require arbitrarily large pipe buffers, or worse, can cause deadlock. *Rc* has syntax for some kinds of non-linear but treelike pipelines. For example,

```
cmp <{old} <{new}
```

will regression test a new version of a command. < or > followed by a command in braces causes the command to be run with its standard output or input attached to a pipe. The parent command (`cmp` in the example) is started with the other end of the pipe attached to some file descriptor or other, and with an argument that will connect to the pipe when opened (e.g. `/dev/fd/6`.) On systems without `/dev/fd` or something similar (SunOS for example) this feature does not work.

Exit status

When a command exits it returns status to the program that executed it. On Plan 9 status is a character string describing an error condition. On normal termination it is empty.

Rc captures commands' exit statuses in the variable `$status`. For a simple command the value of `$status` is just as described above. For a pipeline `$status` is set to the concatenation of the statuses of the pipeline components with `|` characters for separators.

Rc has a several kinds of control flow, many of them conditioned by the status returned from previously executed commands. Any `$status` containing only `0`'s and `|`'s has boolean value *true*. Any other status is *false*.

Command grouping

A sequence of commands enclosed in `{}` may be used anywhere a command is required. For example:

```
{sleep 3600;echo 'Time''s up!'}&
```

will wait an hour in the background, then print a message. Without the braces:

```
sleep 3600;echo 'Time''s up!'&
```

this would lock up the terminal for an hour, then print the message in the background!

Control flow – `for`

A command may be executed once for each member of a list by typing, for example:

```
for(i in printf scanf putchar)
   look $i /usr/td/lib/dw.dat
```

This looks for each of the words `printf`, `scanf` and `putchar` in the given file. The general form is

`for(`*name* `in` *list*`)` *command*

or

`for(`*name*`)` *command*

In the first case *command* is executed once for each member of *list* with that member assigned to variable *name*. If `in` *list* is not given, `$*` is used.

Conditional execution – `if`

Rc also provides a general if-statement. For example:

```
if(cyntax *.c) cc -g -o cmd *.c
```

runs the C compiler whenever `cyntax` finds no problems with `*.c`. An "if not" statement provides a two-tailed conditional. For example:

```
for(i){
    if(test -f /tmp/$i) echo $i already in /tmp
    if not cp $i /tmp
}
```

This loops over each file in `$*`, copying to `/tmp` those that do not already appear there, and printing a message for those that do.

Control flow – `while`

Rc's while statement looks like this:

```
while(newer subr.c subr.o) sleep 5
```

This waits until `subr.o` is newer than `subr.c` (presumably because the C compiler finished with it).

Control flow – `switch`

Rc provides a switch statement to do pattern-matching on arbitrary strings. Its general form is

```
switch(word){
case pattern ...
    commands
case pattern ...
    commands
...
}
```

Rc attempts to match the word against the patterns in each case statement in turn. Patterns are the same as for filename matching, except that / and the first characters of `.` and `..` need not be matched explicitly.

If any pattern matches, the commands following that case up to the next case (or the end of the switch) are executed, and execution of the switch is complete. For example,

```
switch($#*){
case 1
    cat >>$1
case 2
    cat >>$2 <$1
case *
    echo 'Usage: append [from] to'
}
```

is an append command. Called with one file argument, it tacks standard input to its end. With two, the first is appended to the second. Any other number elicits a usage message.

The built-in '`~`' command also matches patterns, and is often more concise than a switch. Its arguments are a string and a list of patterns. It sets `$status` to true if and only if any of the patterns matches the string. The following example processes option arguments for the *man*(1) command:

```
opt=()
while(~ $1 -* [1-9] 10){
    switch($1){
    case [1-9] 10
        sec=$1 secn=$1
    case -f
        c=f s=f
    case -[qwnt]
        cmd=$1
    case -T*
        T=$1
    case -*
        opt=($opt $1)
    }
    shift
}
```

Functions

Functions may be defined by typing

```
fn name { commands }
```

Subsequently, whenever a command named *name* is encountered, the remainder of the command's argument list will assigned to `$*` and *rc* will execute the *commands*. The value of `$*` will be restored on completion. For example:

```
fn g {
    gre -e $1 *.[hcyl]
}
```

defines `g` *pattern* to look for occurrences of *pattern* in all program source files in the current directory.

Function definitions are deleted by writing

```
fn name
```

with no function body.

Command execution

Up to now we've said very little about what *rc* does to execute a simple command. If the command name is the name of a function defined using `fn`, the function is executed. Otherwise, if it is the name of a built-in command, the built-in is executed directly by *rc*. Otherwise, if the name contains a `/`, it is taken to be the name of a binary program and is executed using *exec*(2). If the name contains no `/`, then directories mentioned in the variable `$path` are searched until an executable file is found.

Built-in commands

Several commands are executed internally by *rc* because they are difficult or impossible to implement otherwise.

. [-i] *file* ...

Execute commands from *file*. `$*` is set for the duration to the reminder of the argument list following *file*. `$path` is used to search for *file*. Option `-i` indicates interactive input – a prompt (found in `$prompt`) is printed before each command is read.

builtin *command* ...

Execute *command* as usual except that any function named *command* is ignored. For example,

```
fn cd{
    builtin cd $* && pwd
}
```

defines a replacement for the `cd` built-in (see below) that announces the full name of the new directory.

cd [*dir*]

Change the current directory to *dir*. The default argument is `$home`. `$cdpath` is a list of places in which to search for *dir*.

eval [*arg* ...]

The arguments are catenated separated by spaces into a string, read as input to *rc*, and executed. For example,

```
x='$y'
y=Doody
eval echo Howdy, $x
```

would echo

```
Howdy, Doody
```

since the arguments of `eval` would be

```
echo Howdy, $y
```

after substituting for `$x`.

shift [*n*]

Delete the first *n* (default 1) elements of `$*`.

wait [*pid*]

Wait for the process with the given *pid* to exit. If no *pid* is given, all outstanding processes are waited for.

whatis *name* ...

Print the value of each *name* in a form suitable for input to *rc*. The output is an assignment to a variable, the definition of a function, a call to `builtin` for a built-in command, or the path name of a binary program. For example,

```
whatis path g cd who
```

might print

```
path=(. /bin /usr/bin)
fn g {gre -e $1 *.[hycl]}
builtin cd
/bin/who
```

~ *subject pattern* ...

The *subject* is matched against each *pattern* in turn. On a match, `$status` is set to true. Otherwise, it is set to `'no match'`. Patterns are the same as for filename matching. The *patterns* are not subjected to filename replacement before the '~' command is executed, so they need not be enclosed in quotation marks, unless of course, a literal match for `*` `[` or `?` is required. For example

```
~ $1 ?
```

matches any single character, whereas

```
~ $1 '?'
```

only matches a literal question mark.

Advanced I/O Redirection

Rc allows redirection of file descriptors other than 0 and 1 (standard input and output) by specifying the file descriptor in square brackets `[ ]` after the < or >. For example,

```
cc junk.c >[2]junk.diag
```

saves the compiler's diagnostics in `junk.diag`.

File descriptors may be replaced by a copy, in the sense of *dup*(2), of an already-open file by typing, for example

```
cc junk.c >[2=1]
```

This replaces file descriptor 2 with a copy of file descriptor 1. It is more useful in conjunction with other redirections, like this

```
cc junk.c >junk.out >[2=1]
```

Redirections are evaluated from left to right, so this redirects file descriptor 1 to `junk.out`, then points file descriptor 2 at the same file. By contrast,

```
cc junk.c >[2=1] >junk.out
```

Redirects file descriptor 2 to a copy of file descriptor 1 (presumably the terminal), and then directs file descriptor 1 at a file. In the first case, standard and diagnostic output will be intermixed in `junk.out`. In the second, diagnostic output will appear on the terminal, and standard output will be sent to the file.

File descriptors may be closed by using the duplication notation with an empty right-hand side. For example,

```
cc junk.c >[2=]
```

will discard diagnostics from the compilation.

Arbitrary file descriptors may be sent through a pipe by typing, for example

```
cc junk.c |[2] grep -v '^$'
```

This deletes those ever-so-annoying blank lines from the C compiler's output. Note that the output of `grep` still appears on file descriptor 1.

Very occasionally you may wish to connect the input side of a pipe to some file descriptor other than zero. The notation

```
cmd1 |[5=19] cmd2
```

creates a pipeline with `cmd1`'s file descriptor 5 connected through a pipe to `cmd2`'s file descriptor 19.

Here documents

Rc procedures may include data, called "here documents", to be provided as input to commands, as in this version of the *tel* command

```
for(i) grep $i <<!
...
nls 2T-402 2912
norman 2C-514 2842
pjw 2T-502 7214
...
!
```

A here document is introduced by the redirection symbol <<, followed by an arbitrary eof marker (! in the example). Lines following the command, up to a line containing only the eof marker are saved in a temporary file that it connected to the command's standard input when it is run.

Rc does variable substitution in here documents. The following *subst* command:

```
ed $3 <<EOF
g/$1/s//$2/g
w
EOF
```

changes all occurrences of `$1` to `$2` in file `$3`. To include a literal `$` in a here document, type `$$`. If the name of a variable is followed immediately by `^`, the caret is deleted.

Variable substitution can be entirely suppressed by enclosing the eof marker following << in quotation marks.

Here documents may be provided on file descriptors other than 0 by typing, for example

```
cmd <<[4]End
...
End
```

Signals

Rc scripts normally terminate when an interrupt is received from the terminal. A function with the name of a signal, in lower case, is defined in the usual way, but called when *rc* receives the signal. Signals of interest are:

sighup
: Hangup. The controlling teletype has disconnected from *rc*.

sigint
: The interrupt character (usually ASCII del) was typed on the controlling terminal.

sigquit
: The quit character (usually ASCII fs, ctrl-\) was typed on the controlling terminal.

sigterm
: This signal is normally sent by *kill*(1).

sigexit
: An artificial signal sent when *rc* is about to exit.

As an example,

```
fn sigint{
     rm /tmp/junk
     exit
}
```

sets a trap for the keyboard interrupt that removes a temporary file before exiting.

Signals will be ignored if the signal routine is set to {}. Signals revert to their default behavior when their handlers' definitions are deleted.

Environment

The environment is a list of name-value pairs made available to executing binaries. On Plan 9, the environment is stored in a file system named `#e`, normally mounted on `/env`. The value of each variable is stored in a separate file, with components terminated by ASCII nulls. (This is not quite as horrendous as it sounds, the file system is maintained entirely in core, so no disk or network access is involved.) The contents of `/env` are shared on a per-process group basis – when a new process group is created it effectively attaches `/env` to a new file system initialized with a copy of the old one. A consequence of this organization is that commands can change environment entries and see the changes reflected in *rc*.

There is not currently a way on Plan 9 to place functions in the environment, although this could easily done by mounting another instance of `#e` on another directory. The problem is that currently there can be only one instance of `#e` per process group.

Local Variables

It is often useful to set a variable for the duration of a single command. An assignment followed by a command has this effect. For example

```
a=global
a=local echo $a
echo $a
```

will print

```
local
global
```

This works even for compound commands, like

```
f=/fairly/long/file/name {
     { wc $f; spell $f; diff $f.old $f } |
       pr -h 'Facts about '$f | lp -ddp
}
```

Examples – *cd, pwd*

Program 1 shows a pair of functions that provide enhanced versions of the standard `cd` and `pwd` commands. (Thanks to Rob Pike for these.)

```
ps1='% '     # default prompt
tab='	'     # a tab character
fn pbd{
  /bin/pwd|sed 's;.*/;;'
}
fn cd{
  builtin cd $1 &&
  switch($#*){
  case 0
    dir=$home
    prompt=($ps1 $tab)
  case *
    switch($1)
    case /*
      dir=$1
      prompt=(`{pbd}^$ps1 $tab)
    case */* ..*
      dir=()
      prompt=(`{pbd}^$ps1 $tab)
    case *
      dir=()
      prompt=($1^$ps1 $tab)
    }
  }
}
fn pwd{
  if(~ $#dir 0)
    dir=`{/bin/pwd}
  echo $dir
}
```

Programe 1: *cd, pwd*

Function pwd is a version of the standard pwd that caches its value in variable $dir, because the genuine pwd can be quite slow to execute.

Function pbd is a helper that prints the last component of a directory name. Function cd calls the cd built-in, and checks that it was successful. If so, it sets $dir and $prompt. The prompt will include the last component of the current directory (except in the home directory, where it will be null), and $dir will be reset either to the correct value or to (), so that the pwd function will work correctly.

Examples – *man*

The *man* command prints pages from of the Programmer's Manual. It is called, for example, as

```
man 3 isatty
man rc
man -t cat
```

In the first case, the page for *isatty* in section 3 is printed. In the second case, the manual page for *rc* is printed. Since no manual section is specified, all sections are searched for the page, and it is found in section 1. In the third case, the page for *cat* is typeset (the -t option). Program 2 shows the *man* command.

```
cd /n/bowell/usr/man || {
  echo $0: Manual not on line! >[1=2]
  exit 1
}
NT=n  # default nroff
s='*' # section, default try all
for(i) switch($i){
case -t
  NT=t
case -n
  NT=n
case -*
  echo Usage: $0 '[-nt] [section] page ...' >[1=2]
  exit 1
case [1-9] 10
  s=$i
case *
  eval 'pages=man'$s/$i'.*'
  for(page in $pages){
    if(test -f $page)
      $NT^roff -man $page
    if not
      echo $0: $i not found >[1=2]
  }
}
```

Program 2: *The* man *command*

Note the use of eval to make a list of candidate manual pages. Without eval, the * stored in $s would not trigger filename matching – it's enclosed in quotation marks, and even if it weren't, it would be expanded when assigned to $s. Eval causes its arguments to be re-processed by *rc*'s parser and interpreter, effectively delaying evaluation of the * until the assignment to $pages.

Examples – *holmdel*

Program 3 is an *rc* script that plays the deceptively simple game *holmdel*, in which the players alternately name Bell Labs locations, the winner being the first to mention Holmdel.

```
t=/tmp/holmdel$pid
fn read{
      $1=`{awk '{print;exit}'}
}
ifs='
'     # just a newline
fn sigexit sigint sigquit sighup{
      rm -f $t
      exit
}
cat <<'!' >$t
Allentown
Atlanta
Cedar Crest
Chester
Columbus
Elmhurst
Fullerton
Holmdel
Indian Hill
Merrimack Valley
Morristown
Piscataway
Reading
Short Hills
South Plainfield
Summit
Whippany
West Long Branch
!
while(true){
   lab=`{/usr/games/fortune $t}
   echo $lab
   if(~ $lab Holmdel){
      echo You lose.
      exit
   }
   while(read lab;
      ! grep -i -s $lab $t)
      echo No such location.
   if(~ $lab [hH]olmdel){
      echo You win.
      exit
   }
}
```

Program 3: `holmdel`

This script is worth describing in detail (rather, it would be if it weren't so silly.)

Variable `$t` is an abbreviation for the name of a temporary file. Including `$pid`, initialized by *rc* to its process-id, in the names of temporary files insures that their names won't collide, in case more than one instance of the script is running at a time.

Function `read`'s argument is the name of a variable into which a line gathered from standard input is read. `$ifs` is set to just a newline. Thus `read`'s input is not split apart at spaces, but the terminating newline is deleted.

A handler is set to catch `sigint`, `sigquit`, and `sighup`, and the artificial `sigexit` signal. It just removes the temporary file and exits.

The temporary file is initialized from a here document containing a list of Bell Labs locations, and the main loop starts.

First, the program guesses a location (in `$lab`) using the `fortune` program to pick a random line from the location list. It prints the location, and if it guessed Holmdel, prints a message and exits.

Then it uses the `read` function to get lines from standard input and validity-check them until it gets a legal name. Note that the condition part of a `while` can be a compound command. Only the exit status of the last command in the sequence is checked.

Again, if the result is Holmdel, it prints a message and exits. Otherwise it goes back to the top of the loop.

Discussion

Steve Bourne's `/bin/sh` is extremely well-designed; any successor is bound to suffer in comparison. I have tried to fix its best-acknowledged shortcomings and to simplify things wherever possible, usually by omitting unessential features. Only when irresistibly tempted have I introduced novel ideas. Obviously I have tinkered extensively with Bourne's syntax, that being where his work was most open to criticism.

The most important principle in *rc*'s design is that it's not a macro processor. Input is never scanned more than once by the lexical and syntactic analysis code (except, of course, by the `eval` command, whose *raison d'etre* is to break the rule).

Bourne shell scripts can often be made to run wild by passing them arguments containing spaces. These will be split into multiple arguments using `IFS`, often at inopportune times. In *rc*, values of variables, including command line arguments, are not re-read when substituted into a command. Arguments have presumably been scanned in the parent process, and ought not to be re-read.

Why does Bourne re-scan commands after variable substitution? He needs to be able to store lists of arguments in variables whose values are character strings. If we eliminate re-scanning, we must change the type of variables, so that they can explicitly carry lists of strings.

This introduces some conceptual complications. We need a notation for lists of words. There are two different kinds of concatenation, for strings – `$a^$b`, and lists – `($a $b)`. The difference between `()` and `''` is confusing to novices, although the distinction is arguably sensible – a null argument is not the same as no argument.

Bourne also rescans input when doing command substitution. This is because the text enclosed in back-quotes is not properly a string, but a command. Properly, it ought to be parsed when the enclosing command is, but this makes it difficult to handle nested command substitutions, like this:

```
size=`wc -l \`ls -t|sed 1q\``
```

The inner back-quotes must be escaped to avoid terminating the outer command. This can get much worse than the above example; the number of `\`'s required is exponential in the nesting depth. *Rc* fixes this by making the backquote a unary operator whose argument is a command, like this:

```
size=`{wc -l `{ls -t|sed 1q}}
```

No escapes are ever required, and the whole thing is parsed in one pass.

For similar reasons *rc* defines signal handlers as though they were functions, instead of associating a string with each signal, as Bourne does, with the attendant possibility of getting a syntax error message in response to typing the interrupt character. Since *rc* parses input when typed, it reports errors when you make them.

For all this trouble, we gain substantial semantic simplifications. There is no need for the distinction between `$*` and `$@`. There is no need for four types of quotation, nor the extremely complicated rules that govern them. In *rc* you use quotation marks exactly when you want a syntax character to appear in an argument. `IFS` is no longer used, except in the one case where it was indispensable: converting command output into argument lists during command substitution.

This also avoids an important security hole [Ree88a]. *System*(3) and *popen*(3) call `/bin/sh` to execute a command. It is impossible to use either of these routines with any assurance that the specified command will be executed, even if the caller of *system* or *popen* specifies a full path name for the command. This can be devastating if it occurs in a set-userid program. The problem is that `IFS` is used to split the command into words, so an attacker can just set `IFS=/` in his environment and leave a Trojan horse named `usr` or `bin` in the current working directory before running the privileged program. *Rc* fixes this by not ever rescanning input for any reason.

Most of the other differences between *rc* and the Bourne shell are not so serious. I eliminated Bourne's peculiar forms of variable substitution, like

```
echo ${a=b} ${c-d} ${e?error}
```

because they are little used, redundant and easily expressed in less abstruse terms. I deleted the builtins `export`, `readonly`, `break`, `continue`, `read`, `return`, `set`, `times` and `unset` because they seem redundant or only marginally useful.

Where Bourne's syntax draws from Algol 68, *rc*'s is based on C or Awk. This is harder to defend. I believe that, for example

```
if(test -f junk) rm junk
```

is better syntax than

```
if test -f junk; then rm junk; fi
```

because it is less cluttered with keywords, it avoids the semicolons that Bourne requires in odd places, and the syntax characters better set off the active parts of the command.

The one bit of large-scale syntax that Bourne unquestionably does better than *rc* is the `if` statement with `else` clause. *Rc*'s `if` has no terminating `fi`-like bracket. As a result, the parser cannot tell whether or not to expect an `else` clause without looking ahead in its input. The problem is that after reading, for example

```
if(test -f junk) echo junk found
```

in interactive mode, *rc* cannot decide whether to execute it immediately and print `$prompt(1)`, or to print `$prompt(2)` and wait for the `else` to be typed. In the Bourne shell, this is not a problem, because the `if` command must end with `fi`, regardless of whether it contains an `else` or not.

Rc's admittedly feeble solution is to declare that the `else` clause is a separate statement, with the semantic proviso that it must immediately follow an `if`, and to call it `if not` rather than `else`, as a reminder that something odd is going on. The only noticeable consequence of this is that the braces are required in the construction

```
for(i){
     if(test -f $i) echo $i found
     if not echo $i not found
}
```

and that *rc* resolves the "dangling else" ambiguity in opposition to most people's expectations.

It is remarkable that in the four most recent editions of the UNIX system programmer's manual the Bourne shell grammar described in the manual page does not admit the command `who|wc`. This is surely an oversight, but it suggests something darker: nobody really knows what the Bourne shell's grammar is. Even examination of the source code is little help. The parser is implemented by recursive descent, but the routines corresponding to the syntactic categories all have a flag argument that subtly changes their operation depending on the context. *Rc*'s parser is implemented using *yacc*, so I can say precisely what the grammar is.

Its lexical structure is harder to describe. I would simplify it considerably except for two things. There is a lexical kludge to distinguish between parentheses that immediately follow a word with no intervening spaces and those that don't that I would eliminate if there were a reasonable pair of characters to use for subscript brackets. I could also eliminate the insertion of free carets if users were not adamant about it.

Acknowledgements

Rob Pike, Howard Trickey and other Plan 9 users have been insistent, incessant sources of good ideas and criticism. Some examples in this document are plagiarized from [Bou78a], as are most of *rc*'s good features.

References

[Bou78a] S. R. Bourne, "UNIX Time-Sharing System: The UNIX Shell", *Bell System Technical Journal* **57**(6), pp. 1971-1990 (July-August 1978).

[Ree88a] J. Reeds, "`/bin/sh`: the biggest UNIX security loophole", 11217-840302-04TM, AT&T Bell Laboratories (1988).

EUUG

European UNIX® systems User Group

Autumn '90 Conference and Exhibition

22-26 October 1990

at

Nice Acropolis

Nice

France

TUTORIALS

Monday 22nd October

Registration from 08.00 in the Main Entrance Hall

Tutorial M1 – A comparison of X.11 Toolkits

Tutor: Jamie Watson

This tutorial presents an overview of the most common toolkits available for programming with the X Window System, and a few that are not so common. All information presented is based on the X Window System, Version 11 Release 4.

Intended Audience

The Tutorial is intended for those who are seeking information about the existence and availability of programming interfaces to the X Window System. The most likely attendees are those who are going to have to make a decision on programming with the X Window System, and are interested in learning about the options currently available. Because of the rather general nature of the presentation, the tutorial would also be of interest to those who are interested in learning what the X Window System is, how it can be used, and how to approach programming with it.

Scope

At a minimum, the tutorial will include discussions of:

— Xlib, the "lowest level" library
— Xt, the standard MIT toolkit intrinsics
— Xaw, the MIT Athena Widget Set
— OSF/Motif
— AT & T Open Look
— Sun XView
— InterViews
— Andrew
— Serpent
— Xw
— Xray

Information presented to some degree for each toolkit includes:

— Availability. Where to get it, how much (if anything) it costs.
— Implementation. What language the toolkit is implemented in.
— Portability. What systems does it run on, how much effort would be required to port to a new system.
— User Interface. What, if any, "Industry Standard" is followed.
— Programming Interface. Available programming language interfaces.
— Where possible, simple examples.

Other toolkits and widget sets, such as Poplog,

Continued from previous column

WINTERP, AWL and WsXc are likely to be mentioned in less detail.

Jamie Watson is a software engineer at Adasoft AG, in Switzerland. He has been working with the X Window System since Version 10 Release 3, and has done extensive X Window System consulting and teaching. He also provides distribution in Europe free of charge of the latest release of the X Window System from M.I.T.

Tutorial M2 – Introduction to Object Oriented Programs

Tutor: David Taenzer

This tutorial is targeted for people with experience in traditional programming languages, like C. No experience with object-oriented programming is required.

The tutorial will cover the basic concepts of object-oriented programming. Participants will learn the motivation for object-oriented programming and the basic advantages and disadvantages of this approach. It will discuss five of the object-oriented programming languages available on UNIX platforms: Smalltalk, C++, Objective-C, Eiffel and CLOS (the Common Lisp Object System). It is not however intended as an introduction to any particular language.

The first half of the tutorial will discuss the concepts and features that these (and other commercial object-oriented languages) have in common. These include: object encapsulation, polymorphism and inheritance. The second half of the tutorial will discuss how each of these languages differs from the others. The goals for each language will be described and their unique concepts and features will be discussed in the context of how they match the language goals.

David Taenzer is a member of the Technical Staff in the Advanced Software Technology Group at US West Advanced Technologies in Denver, Colorado. He has over twenty years experience in software development and has been doing research on object-oriented languages, databases and development environments for the past twenty years.

TUTORIALS

Monday 22nd October

Registration from 08.00 in the Main Entrance Hall

Tutorial M3 - MACH

Tutor: Nawaf Bitar

This tutorial is intended for systems developers and technical managers who would like to learn about the Mach operating system.

The tutorial will study the Mach operating system in detail. It will first cover the Mach architecture, philosophy and vision and continue with a thorough study of the three major subsystems (task/thread management, virtual memory and inter-task communication).

Next will be a discussion of the Mach environment and the various user services that are available including the Mach Interface Generator, Network Message Server and Network Memory Server.

Finally, the tutorial will conclude with a presentation of Mach's future direction focusing on the micro-kernel architecture and the dekernelization of UNIX.

Nawaf Bitar is a Research Fellow at the Open Software Foundation where he is part of a cooperating CMU/OSF team developing a Mach 3 based system. Prior to joining OSF he was an engineer in the operating systems group of the Apollo Systems Division of Hewlett-Packard Company where he is involved with the Mach and OSF projects.

Tutorial M4 - Special Topics in C

Tutor: Carol Meier

This tutorial covers a variety of C topics with an emphasis on ANSI changes to the language, portability techniques, and object-oriented programming. It will start with a quick review of C program structure, storage classes and memory model, including the new ANSI const and volatile features. It will cover C preprocessor macros, conditional compilation, assertions and new ANSI directives and operators. It will look at a variety of data structures including multi-dimensional arrays, arrays of pointers, data structures used for accessing command line arguments and environment variables, bit fields, unions, enumerations and the use of typedefs for complex declarations. Students will learn how to use ANSI C function prototypes, write functions that take a variable number of arguments, use pointers to functions and tables of them, use setjmp and longjmp and handle signals.

ANSI C changes to all of these features will be presented throughout the course. Building on these constructs, the tutorial concludes with examples illustrating data encapsulation and dynamic binding, some basic techniques for object-oriented programming.

Outline:

Section I. Memory, Preprocessor
II. Data Structures
III. Functions
IV. Examples — Object-Oriented Programming in C

Intended Audience

This tutorial is aimed at intermediate C programmers who are comfortable with C basics and are ready to explore some of its more powerful features. C programmers interested in ANSI C changes to the language, portability, or object-oriented techniques in C will find these issues addressed in the context of learning the advanced C features listed in the tutorial description.

After completing her B.S. and M.S. degrees in Computer Science at the University of Pittsburgh in 1980 Carol Meier worked as a computer scientist at Bell Telephone Laboratories. From '83 to '88 she worked as an independent consultant presenting hundreds of public and on-site UNIX and C professional development seminars and providing software development and consulting on a contract basis for a variety of companies. She developed and regularly teaches a series of UNIX and C courses for the University of Colorado.

In her current position as Vice-President of XVT Software Inc. a Boulder Colorado company specializing

TUTORIALS

Monday 22nd October

Registration from 08.00 in the Main Entrance Hall

Continued from previous page

in portable user-interface development tools, she is responsible for the implementation of a virtual toolkit that allows applications to be programmed portably across a variety of popular windowing systems including X Windows, Macintosh, Presentation Manager and Microsoft Windows.

Tutorial M5 – A Technical Overview of SVR 4

Tutor: Chris Schoettle

UNIX System V Release 4 (SVR4) continues the commitment for UNIX System V to provide the industry with the basic possible open systems platform. SVR4 is the result of an evolution, unification, and standardization process which introduces new capabilities, improves on earlier releases of UNIX System V and incorporates key features from Microsoft's XENIX system, Sun Microsystem's SunOS system, and the Berkeley Software Distribution (BSD) 4.2 and 4.3 releases while complying with industry standards such as X/Open Portability Guide Issue 3, IEEE POSIX 1003.1 and ANSI X3J11 C.

This tutorial will provide a technical overview of the content and features of UNIX System V Release 4 and will further discuss selected areas in SVR4 internals. The overview will concentrate on the Basic Operating System and will provide additional information on Networking Application Development Environment, User Interfaces, Enhanced Administration and Application Binary Interface. Selected SVR4 internals areas such as Virtual Memory, the Virtual File System and process scheduling will also be addressed.

Chris Schoettle is a member of the technical staff at AT&T UNIX Software Operation (USO) Europe, based in London. He has Bachelor and Master of Science degrees in Computer Science and has worked extensively in the international UNIX systems market. He is currently working closely with major European customers and AT&T USO Development on UNIX System V issues. He has presented the technical content of UNIX System V Release 4 throughout Europe, including the EUUG, Uniforum and AT&T's Software Developer Conferences.

TUTORIALS

Tuesday 23rd October

Registration from 08.00 in the Main Entrance Hall

Tutorial T6 – UNIX System V Performance Tools & Techniques

Tutor: Danny Chen

A survey of the major performance measurement tools available on UNIX system V and a study of several "generic" performance measurement techniques. Topics will include: benchmarking, process accounting and system activity reporter (SAR), the /proc interface and truss, profiling, tracing, a comparison between UNIX SVR4, and new performance measurement interfaces. There will also be at least two in-depth performance measurement analysis and/or modelling case studies.

This tutorial is geared towards application and kernel programmers who are interested in learning how they can measure the resource requirements of their systems. System administrators should also find the discussion of measurement tools and tuning useful. Those already intimately familiar with existing tools may find the tutorial too introductory.

Since joining AT&T Bell Laboratories in 1980, Danny Chen has been working on various aspects of UNIX system performance. He now works in Bell Lab's performance analysis department in Holmdel, New Jersey and serves as a Consultant on UNIX system performance issues for the co-author of two UNIX performance measurement packages that have found their way into commercial use. He has also presented two papers in past Usenix Conferences and has taught a performance measurement tutorial for Usenix and a UNIX system V internals course for AT&T.

He has a M.S. in Computer Science for U.C. Berkeley, a B.S. in Computer Science and a B.A. in Physics from Columbia University and graduated from Stuyvesant High School in New York City.

Tutorial T7 – InterViews

Tutor: Mark Linton

InterViews is a true object-oriented toolkit for the X Window System that emphasizes composition as a way of building user interfaces. The toolkit supports composition of interactive objects (such as scroll bars and buttons) and graphics objects (such as labels, circles and polygons). InterViews is written in C++.

This tutorial uses an example-driven approach to introduce the basic concepts of InterViews and demonstrates how actual applications are built. During the course of the tutorial, a simple InterViews-based application is developed. Attendees should be familiar with C, X and basic object-oriented programming concepts. Familiarity with C++ would be an advantage, but is not required.

PROGRAMME OF EVENTS

TUTORIALS

Tuesday 23rd October

Registration from 08.00 in the Main Entrance Hall

Tutorial T8 – Software Tools for User interface Development

Tutor: Erik Hardy

User Interface life cycle

— requirements definition/analysis
— translation of requirements to design
— development
— maintenance

Special user interface problems

— requirements difficult to specify on paper
— user interface accounts for sometimes >50% of cost over entire life cycle
— maintenance issues
— embedded vs. separate user interface (tight vs. loose coupling)
— accommodation of new user interface technologies over the life cycle

Recent innovations in user interface technology/tools

— accommodates the user interface problems outlined above
— primary emphasis on practical application
— theoretical aspects also covered

Comparison of the different existing tools/approaches

— TAE+ TeleUSE, Serpent, WINTERP are preliminary candidates

Tutorial T9 – Introduction to UNIX — a set of Tutorials for Beginners

Tutor: Prof. Axel Schreiner

Once you have mastered "login" and an editor, this is the course for you.
At about 2 hours each, we will cover the following topics:

```
g/regular expression/p
```

What are regular expressions good for?
How to get some mileage out of programs like "grep"

```
$ awk –F '(print $5)' | sort
```

Who reads RPG or COBOL anyway?

How to generate reports and build useful filters with "sed" and "awk".

```
$ trap 'trap 0; rm –f *; echo sigh' 0 1 2 3 15
```

Why should I use the Bourne Shell?
How to write simple and intermediate shell scripts, hopefully without doing permanent damage to your system.

```
.tm \n% Introduction
```

Can UNIX really replace the common typewriter?

How to write manual pages and simple articles, hopefully containing a few tables, mathematical formulae, and a picture or two.

Recommended reading is "The UNIX Programming Environment" by Kernighan and Pike (Prentice-Hall 1984)

TUTORIALS

Tuesday 23rd October

Registration from 08.00 in the Main Entrance Hall

Tutorial T10 – C++ Programming

Tutor: Tom Cargill

C++ is an extension of C that supports object-oriented programming. To use its new language features effectively programmers must understand how the goals of the object-oriented paradigm can be achieved in C++. The language features are presented in an order that starts by demonstrating improvements in standard procedural techniques and advances to object-oriented programming. The course then explores additional C++ constructs, their implementation, and guidelines for defensive programming. Throughout the course extensive examples are presented which emphasize a practical approach to the language. The role of object-oriented programming and its benefits will be shown.

Outline:

Procedural Abstraction — A Better C
Data Abstraction — Classes
Object-Oriented Fundamentals
Pitfalls — Forewarned is Forearmed
Program Organization
The Implementations of C++

Intended Audience

Attendees will gain an understanding of object-oriented programming and how it is accomplished in C++. Experience programming in C is assumed, as well as knowledge of ANSI C. Familiarity with object-oriented programming is not a prerequisite.

Tom Cargill joined the Computing Science Research Centre of Bell Labs, Murray Hill, NJ in 1981 from the faculty of the University of Waterloo, Canada. His research in Murray Hill involved investigation of debuggers, particularly their user interfaces, which motivated him to start using C++ in 1983. He had some influence on the language in its early years and has continued to use it and follow its growth. He has published accounts of his experience with C++ and given presentations about C++ at numerous technical conferences.

In 1988 he was a member of a joint AT&T/Sun Microsystems team in California investigating the implementation of a new operating system in C++. In the fall of 1988 he moved to AT&T in Denver, Colorado, working mostly on debuggers written in C++. He is now an independent consultant based in Boulder, Colorado. He has a PhD in Computer Science from the University of Waterloo.

Only EUUG National Group or Direct Members are permitted to attend Tutorials.
Each Tutorial lasts for one whole day and will start at 09.30 and finish approximately at 17.30.

PROGRAMME OF EVENTS

Athena Auditorium

Wednesday 24th October

Registration from 08.00 in the Main Entrance Hall

Provisional Technical Programme

Wednesday 24th October

SOFTWARE DEVELOPMENT

10.00
OPENING ADDRESS
Johan Helsingius (SF) — Programme Chair

KEYNOTE 1
Prof. Vic Stenning (Anshar) (UK)

KEYNOTE 2
David Tilbrook (Nixdorf) (CAN)
"Software Hygiene"

Stenning and Tilbrook

Uli Pralle (TU Berlin) (DE)
"Driving the Software Release Process with Shape"

M. O. Pierron & P. Zerlauth (Alcatel Business Systems) (F)
"A Multi-site software Development Environment"

Andy Bartlett (IST) **& Greg Reeve** (White Horse Computing) (UK)
"Lynx — the object-oriented inode-eater"

Tom Christiansen (Convex) (USA)
"A replacement for the Unix manual system"

Ed Gould & Bradley White (Mt Xinu) (USA)
"The 2.6 MSD Software Development Environment"

Tom Neuendorffer (ITC CMU) (USA)
"ATK + 8859 = Multi-Lingual Text and Mail"

Vladas Leonas (Interquadro) (USSR)
"How we survived without the source" — provisional title

Thursday 25th October

Registration from 08.00 in the Main Entrance Hall

Provisional Technical Programme

Thursday 25th October

9.00

Hans Peter Klunder (iXOS) (DE)
"NFS Servers for Filesystems on Optical Disks"

Chii-Ren Tsai (IBM) (USA)
"Potential Pitfalls of Distributed Audit Mechanism"

P. Seghin (Siemens) (BE)
"The XPrint Spooler System"

Kenneth van Wyk (CERT) (USA)
"Computer emergency response — an international problem"

Dan Geer (MIT) (USA)
"Distributed Computing for the Technical Workplace"

Mark Carson (IBM) (USA)
"An X11-based Multilevel Window System Architecture"

STANDARDS AND THINGS
Rob Pike (Bell Labs) (USA)
To be announced

X/Open Speaker (EUR)
"Why XPG"

Johan Helsingius & Jaana Porra
(Penetron) (SF)
"Open Systems and the Research Community"

Pamela Gray (Marosi) (UK)
"Why Factionalism?"

Unix International speaker
"Road map" — provisional title

OSF Speaker
"The OSF/1 Operating System — An Open Computing Base for the 90s"

STANDARDS BoF,
including presentation from Dominic Dunlop

Friday 26th October

Registration from 08.00 in the Main Entrance Hall

Provisional Technical Programme

Friday 26th October

09.00 OSI

Hans Georg Baumgartel (SoftLab) (DE)
"X.400 Server Toolkit on UNIX"

Andrew Findlay (Brunel University) (UK)
"Setting up an X.500 Directory Service"

Steve Kille (UCL-CS) (UK)
"Integration of Message Handling and Directory — A System Manager's View"

UIMS

Alexios Zavras (Univ Wisconsin) (USA)
"Extending 4.3BSD Unix to support Greek characters"

Erik Hardy & Daniel Klien (SEI CMU) (USA)
"The Serpent UIMS"

Zoltan Hidvegi (Videotron Development Inst) (H)
"User Interface Builders: a Compromise platform in the User Interface War"

Mike O'Dell (Bellcore) (USA)
"Putting UNIX on very fast computers"

Peter Snyder (SUN) (USA)
"tmpfs — A virtual memory file system"

Istvan Szeker & Zoltan Vinceller (Eotvos Lorand Univ) (H)
"UNIX pipe extension on network for creating distributed functions"

Sam Leffler (USA)
"Fun with Fax"

William Roberts (QMW) (UK)
"The more I find out, The less I know? NFS fileserver benchmarks (including one that works!)"

FINISH and See you in Tromsø

How to Book

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

European UNIX system User Group
Central Office
Owles Hall, Buntingford
Hertfordshire SG9 9PL, UK

Tel: + 44 763 73029
Fax: + 44 763 73255
email:euug@EU.net

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 can only be taken when paying by credit card.
The EUUG Secretariat will acknowledge your bookings by sending you a recepted invoice together with further details for registration.

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

Payments may be made by 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
   61 Grassmarket
   Edinburgh
   Scotland EH1 2JF

   Account Number: 00613997
   Bank Sorting Code: 80-31-50

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

3. By VISA, ACCESS, EUROCARD or MASTERCARD, with card details appearing on the Booking Form.

   NOTE: Closing date for all bookings is 17th October 1990.

Costs

TUTORIALS

Tutorial per person if booked before	1st August	Members Only £205
Tutorial per person if booked after	31st July	Members Only £305
*Tutorial per person if Registration and Payment on the door		Members Only £405

CONFERENCE

3 Day Conference if booked before	1st August	Members £210 Non-Members £290
3 Day Conference if booked after	31st July	Members £305 Non-Members £385
*3 Day Conference if Registration and Payment on the door		Members £405 Non-Members £485

*These can only be accepted if space allows.

Booking Form for Conference and Tutorials

Please complete this form and send it, with cheque or evidence of payment, to **EUUG Central Office, Owles Hall, Buntingford, Herts SG9 9PL, UK** (Block capitals please). Please note that forms sent without cheque or evidence of payment will be returned to you unregistered.

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 22nd October . . . £ ________

Please reserve me a place for Tutorial No ________ on Tuesday 23rd October . . . £ ________

Do you require Vegetarian meals? Yes ☐ No ☐

Extra ticket for Social Event — £30 each ☐ . . . £ ________

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

TAPE Please reserve me a copy of the Conference Tape **at £60** ☐ 1600 bpi, 1/2" reel tapes . . . £ ________
☐ 1/4" QIC-24, cartridge . . . £ ________

Please read the section **"HOW TO BOOK"** Total £ ________

PAYMENT METHOD

☐ UK Cheque, Banker's 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. EUUG ***must*** *receive the actual amount due.*

☐ by **VISA**

☐ by **ACCESS/EUROCARD/MASTER CARD**

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

Address of cardholder ______________________

____

____

Card Account No. ______________________ **Date of Expiry** ______________________

Signed ______________________ **Date** ______________________

An Eastern View of the EUUG Spring Conference in Munich

Guenther Fischer
gf@tu-k-ddr.cs.tu-berlin.de

My EUUG-Story starts in November 1989 as the wall breaks down. But stop - first some words about our UNIX history.

I have been working at the Technical University in Chemnitz (until now Karl-Marx-Stadt) in the department of Informatics since 1980. We started with UNIX in 1983 and our first goal was to write a C Compiler for a GDR UNIX System written totally in assembly language for an IBM 360/370 like machine. This system, called PSU, worked and still works as a subsystem under operating systems like OS MVT & TSO and so on. We succeeded after half a year and so we started to gain experience in C programming, porting software and "portable" software.

Oh, it was terrible this UNIX system ‘‘thought’’ in EBCDIC code - portability of version 7 sources often broke down. In the following years we also worked in the area of the UNIX kernel, first in the area of education, second in the research field. Surely "research" is a big word, but we lived on the other side of the wall and so information, systems and sources came rarely and often not in a legal way. We wrote device drivers, made changes to the kernel and library to be happy with the standards.

In the GDR there is also a UUG called EAG for developer and user group. This first started only with the gurus and later with all interested users to bring the ideas over. Stupidly it was impossible to contact our colleagues from the GUUG, but sometimes (if we got one) we read their newsletters, so Snoopy (Sebastian Schmitz) was well known to us.

So I wrote a letter to Snoopy in November to make a first contact and he helped me with an article in the GUUG-News and with a mailbox. In the beginning of December I made to a short trip to Munich with my family. At the same time I made a short visit to Anton Gerold from the GUUG and had a nice talk with him.

So I heard about the EUUG Spring Conference in Munich and the possibility of getting a Student Grant for the conference. With the help of Snoopy, Hans Strack-Zimmermann, Ernst Janich, Helen Gibbons, ... I have got the grant and now I have to write this article. Thanks to all people for helping me, also to Thomas Habernoll from the TU Berlin (West), who gave me a first possibility for e-mail - but this is another story. Often it works, always better than telephone or yellow mail - German post cars are yellow.

I don’t want to finish my introduction before making a remark about my English: this was my first conference held in the English language - but I have read papers in English for many years, but there was no possibility of listening and speaking. So I have asked Ernst for writing this as a second report about the conference - My "eastern" view of the EUUG Spring Conference in Munich.

Sunday

After breakfast I started the trip to Munich in my car. At 2 p.m. I was in the Sheraton hotel (very noble, much expensive) - there was the EUUG governing board and I wanted to meet Ernst. A first misunderstanding - I thought I have got the grant for conference and tutorials, but I had got only the conference. So Ernst tried to get the Monday tutorial M3 "Chorus" for me. He succeeded, but I have to write a second report now.

So I quickly forgot the tutorial on Tuesday - Munich is a nice city and a third report would be terrible.

After my short talk with Ernst I was driven to my friend - a colleague who had left the GDR one year ago - where I could stay this week. It was late this evening - much to talk about and so many television programs.

Monday

I was driven by car to Sheraton every day. To drive by car in Munich is not as difficult as I thought, but to get a place to park is terrible - thankfully the Sheraton is outside the inner city. Public transport in Munich was much too expensive for me. You must know my University could give me only the time off and till now I could exchange only 200 DM for 600 GDR-Mark for one year.

In the Sheraton at the EUUG-Registration I have got much material - a nice bag, an EUUG pencil engraved with my name, the conference proceedings and the printed slides of the tutorial. I think this is a very good service.

It is 8 o'clock in the morning - the tutorial started at 9.30 a.m., so I have time to study my material - proceedings, sightseeing information about Munich, an invitation card for the conference dinner on Thursday in the Hofbrauhaus and much more.

At the beginning of the tutorial we were checked in. You may remember that Lenin said: "To trust is good, to check is better". You could see it on your badge, which you also got at Registration too. On the badge you could find your name, your institution and "C" for conference, M[1-3] and T[1-3] for tutorial on Monday and Tuesday, if you have paid for them or you have got a grant.

About the tutorial itself I don't want to report in detail, because there was a Chorus talk at the conference and perhaps I have to write a second report about this tutorial. I would have liked to go to all 3 tutorials M1 Programming with OSF/Motif - I have colleagues who are working in the area of user interfaces, M2 Advanced Network Programming - this is the area I have to deal with in the near future and M3 Chorus - the area of operating systems is one of our main aspects in education and research that we have dealt with in the last few years and now we are looking for ways to redesign the kernel in an object oriented manner. So we have interest in systems such as Chorus, Mach, BirliX, System V Release 5 (rewritten in C++), ...

At 12 o'clock is lunch time. For a moment I was afraid, lunch at the Sheraton would be too much expensive for me, but naturally it was included in the conference and tutorial fee. A very good lunch - you could have what you wanted. Another important aspect of such a meeting: you can meet many old friends and make some new ones. I was glad to see the unido crew with Anke Goos, Torsten Beyer and Michael Kutschke, who had visited Chemnitz some weeks ago and all the others who helped me to be here.

Tuesday

No tutorial - no report. One eye with tears - the other with fun. I will make a walk through Munich - some sightseeing, some shopping, but the weather is terrible - a cold and rainy day. I think I will see Munich again in summer.

Wednesday

The conference is starting at 9 a.m. - a good time. There are much more people than during the tutorials - I think 400-500. The conference is in a large ballroom of the Sheraton, a good environment, with all technical conditions needed. The keynotes came from Richard F. Rashid about the Mach system. The representation is excellent - from slides to video, I can understand and follow the talk and he makes some jokes. There are many problems solved, some that we have dealt with too: restructure of the kernel to fight against complexity and to solve the problems of distributed and multiprocessor systems, maintain machine independency - there is a single machine independent module, and so on. There was also a nice talk about the name Mach. From the suggested names Melange, Moose, Koma and Muck the discussion - I think during the coffee breaks - evolves to the name Mach.

After the keynotes there was a coffee break in the area of the exhibition. (Coffee break always at 10.30 a.m. and 4 p.m. - a good thing for coffee junkies.) The exhibition was very interesting for me too (I couldn't go to the CeBit) and I had have some interesting talks. The giant IBM with its new RISC System 6000: very interesting, good presentation, much show business.

There are many books around about UNIX - you could buy some. I couldn't.

SUN of course very interesting for GDR universities because of a new SPARC station costs under 10000 DM - I hope we can buy one in the near future and COCOM will fly away for ever.

ICL with the reference port of UNIX System V Release 4 on a SPARC system. I had have a nice talk and the possibility to look inside the system. System V, BSD/SUN and XENIX all in one and conform to POSIX - is that possible? It seems to be, but some questions remain.

Furthermore, I visited the BSP Presentation, because we have some experience with SCO Systems.

Sorry I came late from coffee break. A man with long hair (perhaps one from the 60's) from the Chorus team spoke about Multithreaded processes in Chorus/Mix. Another interesting design of a modern operating system structure. Parts of the kernel (the process manager) are written in C++.

Then Herman Moon gave an overview about DACNOS - I hope we will never get such problems: VM/CMS, VAX/VMS, PC-DOS, OS/2 and UNIX in one environment - terrible.

Then Ian Elliot spoke about the AIX Version 3 kernel of the new IBM RISC. He spoke about memory management for the new system. 52 bits for virtual addressing gives you the freedom to map all files in virtual memory, no buffer cacheing is needed, file system inconsistency *couldn't* happen. Shortly after that, the words "under normal circumstances" were added. All in all a good IBM presentation - no questions afterwards.

Then lunch - in the same fantastic environment. But you can get much more service at the conference: The face saver - a video camera coupled with a computer saves your face in digitalised manner. Sometime later you can get a copy and you can get the picture via EUnet. Somebody told me there is a vwho command, where you can see the people logged in in a visual way. Perhaps you can find my face saver picture on the body of this report. Another good service is the TCP/IP connection to unido - I think a 64 KBit line. So I could send a mail to my colleagues in Chemnitz and Torsten shows me a ftp session to the MIT - it works well. My compliments to people who installed the service.

Jump to the afternoon after the coffee break - never forget the coffee break. Some talks about network services. First Gabriele Cressman-Hirl from SUN - one of some Unixies at the conference - a talk about Network Management Gateways. I think there are more Unixies in the west of Europe than in the east in relation to opposite sex.

Then Toshiharo Harada from Japan about PCSERVE: An attempt to integrate PC users into the UNIX Community. PC users or better MS DOS users can get UNIX services like mail or news in the MS DOS environment without knowing about UNIX. After the interesting talk there came a question: "Can I get the sources?". Toshiharo said: "Yes, of course, I have written it by myself". After a short time: "Oh, I think I have to ask my boss". Laughing everywhere ...

In the evening there is a party in the rooms of the iXOS Software Company. At 6 p.m. two Shuttle Buses were starting out to iXOS. In the bus we get a special badge for the party - only 50 or 60 people can get in. What fun that I am one of them. iXOS is up to date: on my badge you could see TU Chemnitz - the decision to renaming Karl-Marx-Stadt to the old name Chemnitz was only two days old.

A very nice party: much to eat, much to drink and all in a wonderful environment - thanks to Hans Strack-Zimmermann and Eberhard Faerber, the bosses of iXOS for the invitation. I have seen 3 rooms of the company and I will give a short description. First the room, where you could get something to drink, called X-Teria, I think. It's also a good place to have a talk with colleagues or guests, if there is no party - it could happen sometime. The second room has about 5 to 7 places to work - all of them with computers or terminals naturally. Isn't it too noisy? Don't worry, take the CD walkman you have got from iXOS and you have your own acoustic environment. What a service!

The third room is for teaching - 10 places with 5 terminals and an overhead display for the teacher.

At a good time the Shuttle Bus took us to the Sheraton, where my car was waiting.

Thursday

Till lunch there are 2 talks, which have my special interest: Nick Williams - The Educational On-line System EOS. It enables students and lectures to exchange course materials. He spoke about the development of the user interface, from simple shell scripts to an X-window interface.

I am more interested in the facts of the changing education environment than the implementation details.

Then Dieter Konnert about Dynamic Driver Loading for UNIX System V. Because I've implemented drivers by myself, I know how terrible it is to have to relink the kernel and reboot the system for every single test. I think it could be a good tool for driver developers.

Lunch time - all is well, time to tell about a another service. There is a Daily Newsletter during the conference. You could find some information and remarks around the conference. I've got only the Friday-Newsletter - I forgot to look for it on the other days.

In the afternoon there is an interesting talk with big bosses of the German computer industry: Georg Faerber (first PCS, now Mannesmann/Kienzle), Bernd Woebker (Nixdorf, this company is getting married with Siemens now) and Klaus Gewald (Siemens).

This talk was moderated by Hans Strack-Zimmermann. First they gave some statements and then there is a discussion. The most interesting part came from Snoopy. He had so much to tell that he spoke like a waterfall. His boss Mr. Strack-Zimmermann must stop him. One point of the discussion there is a comparison of a big computer company and an Elephant. The Siemens manager said: "Elephants are very sympathetic". Strack-Zimmermann: "But gray, always gray".

At the end of the discussion the moderator asks the other three: "If you have to control an atomic power plant, would you decide to take OSF/1 or System V Release 4 as the operating system?".

How would you decide? - I think that no atomic power plant at all would be the safest.

In the evening came the most important session of the conference: Conference Dinner in the "Hofbrauhaus am Platzl". I would say: One time in your life you should go there - but one time is also enough. You can listen and look to typical Bavarian culture (folk music, folk dance (Schuhplattler) and so on). I think it is mostly for foreign tourists especially from United States and Japan. This is the view they often want to have from the Bavarians (or Germans?). At the beginning there were introduced the new EUUG Executive Board with the chairman Michel Gien. Some people, who have done something for the EUUG have got some presents like a clock with the EUUG logo.

Afterwards - or was it before? - we surrounded to the tables which were filled with typical Bavarian specialities, a variety af salads and other choices - thanks to Mrs. Gerold, who has arranged the menu. Naturally there was Bavarian Beer to drink - 4 sorts, light and dark. You could have the "normal" litre glasses and small glasses - this can only be normally for professional drinkers.

Shortly after 10 o'clock I left the dinner, so I couldn't listen to the EUUG Song Contest in Spring of 1990 - could only read about it in the Friday Newsletter.

Friday

The highlights of the Friday morning: Snoopy's Britta and child. They came during the first talk by Ian Chapman about "The Design of a PC-based NFS Client from DOS Programmers Perspective". Siemens's Distributed File System DFS, a client-only NFS implementation for MS DOS was introduced. After that, Snoopy's talk. He started with an official introduction of his 16 month old child - Snoopy++ or better Britta+Snoopy. Then he spoke about "Archiving Documents on WORM Disks - the NFS Approach". This was a report about some work iXOS has done in contract to Siemens. WORM disks - Write One, Read Many - are now ideal media for archiving because of their long lifetime (10 years at least), high capacity and acceptable transfer rates. Sometimes Snoopy++ seems to be in opposite opinion - Snoopy++ cried, so Britta with child left the conference room.

Jump to the last coffee break. Thomas Herz from Interface Connection GmbH told about "HIT-Multicode - Implementation of a multilingual Word Processing System" - a very good demonstration of internationalisation in text processing. You can switch from one language to another in one text. No problem - but it works also from Arabic to English and vice versa - remember Arabic goes from right to left. Then George M. Taylor about an X-Windows Teletext Service for UNIX. He is a Scotsman and I saw him always with his kilt as he went to the Conference Dinner. Last talk before last lunch Rudolf Koster about "Porting Banking Software from VMS to UNIX". UNIX in the banking area, who would have thought it a few years ago?

After lunch Pascal Beyls from France told about "Is UNIX resistant to computer viruses". They have made some experiments with viruses in the UNIX area. In the future the danger of getting a virus will increase because a binary program exchange is in progress and distributed systems at computing level will be coming. On the other side we will have some new security features in the near future - OSF/1 will get the security stage B1 of the Orange Book in September 1990 and AT&T will release new security features for the V.4 system at the end of the year. With such optimistic views into the future I will stop my report.

In conclusion I can say: If you can get the money come to the next conference - much to learn, many interesting people, meet and get friends and all in a good environment.

Thanks to the EUUG and all who have done the work.

UNIX Training from the Market Leaders

- ☐ UNIX: A System Overview
- ☐ UNIX Basics for Users
- ☐ UNIX for System Administrators
- ☐ UNIX Fundamentals for Programmers
- ☐ Advanced UNIX Tools
- ☐ Advanced Programming in the UNIX Environment
- ☐ Advanced UNIX System Administration
- ☐ UNIX Device Drivers & Kernel Overview
- ☐ STREAMS under UNIX System V.3
- ☐ UNIX System V.3 Kernel Seminar for Source Licencees

For further details of our UNIX Curriculum (or training in C, C++, INFORMIX and ORACLE) please photocopy this page and return it to the address below. Or call Karen Hall on +44 71 251 2128.

- ☐ Please send me more details about the ticked courses
- ☐ I would like more information on your other training curriculum.

Name ____________________

Company ____________________

Address ____________________

____

THE INSTRUCTION SET

The Instruction Set, City House, 190 City Road, London EC1V 2QH, UK
Tel: +44 71 251 2128 • Fax: +44 71 251 2853

London Conference Report

Maggie Cooper
m.cooper@city.ac.uk

City University

Maggie Cooper is a computational linguist, currently working at City University, London, on an Esprit project "Interactive Dialogues for Explanation and Learning" (IDEAL). Having been exposed to UNIX at an impressionable age, she regresses to a primitive Systems Creature at times of crisis (if you've ever been to City ...). She is also known as a renegade Time Lord (rani@tardis.ed.ac.uk).

Sunday

The Conference began for me at 10 a.m. on Sunday, when helpers met at the Royal Lancaster to put notes into tutorial folders, fill delegate bags, etc. As there were about fifteen of us, this was accomplished in record time. A further record was broken with the arrival of the bill for coffee and mineral water! We lunched out, having left with a good selection of freebies. The Conference Suite was not available to us until 2 a.m. on Monday, and so people reassembled then to set things up. Luckily as I live in Greater London rather than the Inner City, I was exempted from this.

Monday

Although it was an early start in comparison with my normal working hours, getting in from Wimbledon was reasonably painless. I'd opted to help out at Berry Kercheval's "Introduction to X Concepts" tutorial. As it turned out, there was very little for me to do, which was just as well, because after working through the basics, he began on the details of the more arcane Xlib functions, where I got lost a couple of times. I've only had very limited experience with X, but managed to learn a great deal. The tutorial notes and overheads are extremely well prepared, so I'll be able to work over the more complicated bits again. The hotel provided us with jars of sweeties and bottles of mineral water, in case we began to wilt. At lunch I shared a table with people from other tutorial groups, which was a good opportunity to discover what they'd been doing.

Tuesday

I thought since I'd made it "in good time" yesterday, i.e. with ten minutes to spare, I'd catch a later tube. Not such a good idea, but I managed to get in just as the tutorial was about to start Paul Kimball's "Programming with the X Toolkit Intrinsics". A practical introduction to programming with X Toolkits, with plenty of examples, this followed on very well from the previous day's tutorial which had provided me with enough background information to cope relatively easily. I'm rather proud that I've since had great success with a customization of one of Paul's handout sample programs. Again, the tutorial notes are of a very high standard.

Wednesday

Since a friend was staying with me for the conference proper, it was an even earlier start, to make sure he had time to register. I really needed a cup of coffee to get me started, but luckily Sunil

Das's opening address was funny enough to wake me up. Not all the humour was intentional
he was totally confused as to whether the Tower of London or Tower Bridge was the venue for the conference reception.

Rob Pike's keynote speech was the first announcement of "Plan 9 from Bell Labs". Basically, UNIX isn't really fun any more; it's too big, too standardized, too immutable, where standardization has discouraged sensible system building. Plan 9 is a distributed system which offers an alternative to the increasingly ubiquitous trend of workstations connected by local area networks. He puts forward as a reasonable model individual bitmapped terminals with (high power) computing available from servers over a network. I'm aware that people are already realizing that the workstation model is uneconomical, and the move into servers will gain momentum. The goal in building Plan 9 is to have one timesharing system for all of Bell Labs. I was very interested to note the emphasis put on ease of use from home. A stills demo clarified some of the features of the system. Security was carefully considered; there is no superuser as far as the filesystem is concerned, and administrators don't have access to user's files. Plan 9 is currently for local use, but they hope to distribute it in the future. David Presotto and Tom Duff's presentations gave further details about the system, concentrating on multi-processor streams, and the shell.

After lunch (and a fair amount of wine) Dennis Richie's talk looked at the ANSI standard for C, and possible extension. Contrasting with the earlier speakers, he was for standards, with a few reservations. The next talk by Ken Thompson discussed a new C Compiler. As I realized it was likely to be a little too technical for me, I opted to watch it on the very useful monitors outside the main conference room. Tom Cargill then gave an well-argued paper on multiple inheritancy. He didn't say it was unnecessary, just that he'd not seen a C++ program where it was properly used, and that those who want it in the standard should justify it. I think most people were prepared to agree that a standard should be as small and as general as possible, and that those things which are already working practice should be standardized.

At this point I had to leave to deal with some committee type nonsense back at City. I wasn't back in time to join the BoF session at which Chris Torek's splendid daemon teeshirt was first seen and much coveted, but luckily I didn't miss any of the cocktail party. The supply of drink was impressive, and the band livened things up considerably. The real highlight of the evening came with Karels and McKusick happily debugging 4.4 BSD over Internet while the band played on!

Thursday

The aforementioned impressive supply of drink was responsible for my late arrival. Things weren't improved when I found that e-mail wasn't working. The laser infra-red link brilliantly improvised to make up for British Telecom's shortcomings was distorted by the hot weather, and only works during the evening. I was in time to catch most of Greg Rose's talk about limits. It's interesting that the approach to user security is similar to that of Plan 9, with the sharing out of superuser privileges.

Next came the Berkeley System Daemons slot, with McKusick and Torek looking at lots of interesting BSD internals for the future. Again, I wasn't able to follow everything, as my background knowledge is insufficient.

After coffee, Jon Bentley gave us a talk on "Pictures of Programs", which came across very well despite problems with some of the multimedia stuff. This was followed by Michael Hawley's talk on symphonic emulation, which I specially enjoyed. I suspect that quite a few in the audience (including myself) were unable to tell what was wrong with the Beethoven he played.

Daniel Klein was unable to present his paper for reasons of health, and Jim Reid stood in for him, occasionally adding his own comments on password security. Although the advice given isn't new, the paper contains some interesting statistics. Greg Rose's paper was pretty much a sales pitch for Supercomputers, originally written for a Supercomputer conference. Forsyth's talk on kernel implementation was very technical. He acknowledged that he took some stuff from EMAS. As I still have strong Edinburgh links, I can confirm that it's still up and running there. (Wonder if they'll thank me for that?)

The standards talks generated quite a bit of debate, but I think most people felt that as a codification of existing practice they make sense.

People then made their own way to Tower Bridge for the Reception. Some of the routes adopted for this journey showed great originality, and there was talk of yet another conference competition. Both Bridge walkways were hired for the occasion, and while it was easy to catch the person you wanted to talk to, it was difficult to stand out of people's way. There appeared to be rather a scrum for the buffet and drinks, too.

Friday

Hmmm. Another late start which I don't think I'll bother to explain. However I caught most of Andrei Yaneff's talk about G3 and 3-D graphics. It looked rather difficult to use, with a highly detailed syntax. Next Brian O'Donovan gave a very good presentation on distributed RCS.

Stuart Feldman's paper on large-scale software development under UNIX spelled out some basic facts concerning politics, sociology, and engineering. Brian Kernighan continued this theme (after showing photographs of a wide range of things called UNIX). He pointed out that productivity is far higher now than twenty years ago, and looked at how the evolution of UNIX has contributed to this. I was interested in his comments on the little languages and how he sees them as being in some ways the most characteristic feature of UNIX; much of my time is spent playing with them.

After lunch I decided to check out the FaceServer the result was not as bad as some I saw during the conference, though I don't think I'll incorporate it into my CV. As by now the conference team were beginning to wind up the desk etc., I wasn't able to catch the next papers. Everything was dismantled by the teabreak, and so I got to hear Andrew Tanenbaum's talk on Amoeba, a new system for distributed computing. His comments on abandoning the workstation model brought to mind the keynote speech.

The closing address was given by Mick Farmer, who also announced the results of the conference competition to invent a new library function/system call, and to devise the worst insult for X. In the event, the competition was judged on the insults for X alone, and amazing prizes were distributed. See his column for the details.

INTEROP 90

8-12 October 1990
San Jose
California
USA

A large conference,
tutorials and tradeshow dedicated to
networking and interoperability.
Contact Interop, Inc. for more details:

TEL: +1 (415) 941-3399
FAX: +1 (415) 949-1779

TEST TOOLS FROM THE WORLD LEADERS IN CONFORMANCE TESTING

BSI QUALITY ASSURANCE

POSIX STANDARDS TESTING

As world leaders in conformance testing we are pleased to make available the test tools we use in formal software validation:

POSIX

The IBM POSIX Conformance Test Suites (PCTS) tests an operating system environment for conformance to the POSIX.1 and POSIX.2 standards. The suites are designed not only to help you measure how closely an operating system conforms to the Institute of Electrical and Electronic Engineers (IEEE) 1003.1 and 1003.2 POSIX standards but also to help you determine what changes are necessary to an existing operating system to make it conform to these standards. Thus, the 1003.1 PCTS and 1003.2 PCTS are useful as development tools as well as conformance measurement tools.

C

The Plum Hall Validation Suite for C tests a C language translator for conformance to ANSI X3.159-1989 (ANSI C). The suite is also configurable to allow testing against earlier de-facto standards such as K&R and SYS V. The suite is useful and applicable to compiler implementors and users, especially those involved in the procurement of C compilers. The Plum Hall Validation Suite for C is the chosen test tool for the European Conformance Testing Service for C (CTS-C).

The IBM Posix Conformance Test Suites (PCTS), and the Plum Hall C Validation Suite (CVS) are available from BSI Quality Assurance at the following prices:

	(PCTS)		(CVS)	
☞	£ 2250	Commercial Establishments	£ 10000	Full Suite
☞	£130	Educational Establishments	£ 7500	Conform only

For further information and order form please complete the attached slip and return to:

Software Engineering Department
BSI Quality Assurance
PO Box 375
Milton Keynes
MK14 6LL

Email:	**sed@bsiqa.co.uk**	
Tel:	**(0908)220908**	**+44 908 220908**
Fax Gr 2/3	**(0908) 220671**	**+44 908 220671**

I would like more information on the IBM Posix Test Suite and/or the Plum Hall Test Suite (please delete as appropriate).

✂

Name: ....

Address: ....

....

....

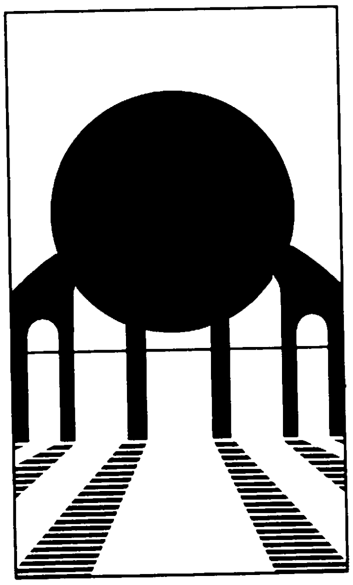
Phone: ....

EUUG

European UNIX® systems User Group

PRELIMINARY ANNOUNCEMENT and CALL FOR PAPERS

EUUG Spring '91 Conference and Exhibition at



Tromsø, Norway
20-24 May, 1991

Preliminary Announcement

The NUUG will host the Spring '91 European UNIX Systems User Group Technical Conference in Tromsø, Norway, Europe.

The Conference will concentrate on Distributed Open Systems in Perspective. In addition to an overview of the issues involved in the design of distributed open systems the conference will address problems encountered and solutions found when distributed open systems are employed.

The three day Conference with commercial Exhibition on Open Systems, UNIX and related subjects will be accompanied by Technical Tutorials on Monday 20th and Tuesday 21st May, followed by the main conference on Wednesday 22nd to Friday 24th May.

A pre-conference registration pack containing detailed information will be issued in February, 1991.

Call for Papers

The EUUG invites papers from those wishing to present their work. Full papers or extended abstracts must be submitted. All submitted papers will be referred to be judged with respect to their quality, originality and relevance.

Suggested subject areas include, but are not limited to:

★ ***The Applicability of Distributed Open Systems***
  - ★ ***Security***
  - ★ ***Reliability***
  - ★ ***Transparency***
  - ★ ***Interoperability***

★ ***Distributed Applications***
  - ★ ***Tools***
  - ★ ***Heterogenous Distributed Environments***
  - ★ ***Distributed Databases***
  - ★ ***User Interfaces in Distributed Environments***

Important Dates

Abstract deadline	5th November, 1990
Acceptance notification	17th December, 1990
Final paper	4th February, 1991
Closing date for Student Grant Application	11th February, 1991

Method of Submission

Full papers or extended abstracts must be submitted by post to the EUUG Secretariat and, if possible, in electronic form to euug-tromso@eu.net. All submissions will be acknowledged by return of post.

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 Executive as soon as possible.

Additional Information

We will be pleased to provide advice to potential speakers. We can be contacted at the addresses 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 EUUG central office.

Useful Addresses

Secretariat
EUUG
Owles Hall
Owles Lane
Buntingford
Herts
SG9 9PL
UK

Tel: +44 763 73039
Fax: +44 763 73255
Email: euug@Eu. net

Conference Executive
Dr. Ernst Janich
Nixdorf Computer AG
Söflinger Str. 70
D-7900 Ulm
West Germany

Tel: +49 731 1526 464
Fax: +49 731 1526 105
Email: janich@nixulm.uucp

Programme Chair
Frances Brazier
Vrije Universiteit
Vakgroep Funkheleer
subf Psychology
de Boelelaan 1115
1081 HV Amsterdam
The Netherlands

Tel: +31 20 5483885
Fax: +31 20 426275
Email: frances@psy.vu.nl

Local Organisation
NUUG
Magnar Antonsen
University of Tromsø
Dept. of Computer Science
N-9000 Tromsø
Norway

Tel: +478344043
Fax: +478344580
Email: magnar@sfd.uit.no

Tutorial Executive
Neil Todd
GID Ltd
1 Captains Gorse
Upper Basildon
Reading
Berkshire RG8 8SZ
UK

Tel: +44 491 671 964
Email: neil@gid.co.uk.

Programme Committee
Donal Daly
ICL Training Services
5 South County Business Park
Leopardstown
Dublin 18
Ireland

Tel: +353 1 956644
Fax: +353 1 956555
Email: donal@ts.icl.ie

Hans W. Stack Zimmermann
iXOS Software GmbH
Technopark
Bretonischer Ring 12
D-8011 Grasbrunn
West Germany

Tel: +49 89 46 10 05 0
Fax: +49 89 46 10 05 99
Email: unido!ixos!hsz

Dag Johansen
Dept. of Computer Science
University of Tromsø
N-9000 Tromsø
Norway

Tel: +47 83 44047
Fax: +47 83 44580
Email: dag@sfd.uit.no

For further details, contact:
EUUG Central office:

European Unix® systems User Group
Owles Hall, Buntingford, Herts SG9 9PL, UK.
Phone: (+44) 763 73039 Fax: (+44) 763 73255 Network Address: euug@Eu.net

EUUG Executive Report

Helen M W Gibbons
euug@EU.net

European Unix system Users Group (EUUG)

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

The New Executive Committee

The new EUUG Executive Committee met for the first time after the Governing Board meeting on the 19th May and immediately set itself the task of reforming more efficiently and tackling the organisation of new services.

To tackle this a series of sub committees have been set up, each with an Executive Committee member in charge and one shadow assistant whose job it is to help with the assigned portfolio should the person in charge be away, or for any reason, be unable to undertake the work. Initial portfolios have been assigned as follows:

- **Administration and Finance**

Exec in charge	–	Nigel Martin
Assisting	–	Kim Biel-Nielsen
S.C. members	–	Ernst Janich
	–	Helen Gibbons

- **Co-ordination with National Groups**

Exec in charge	–	Norman Hull
Assisting	–	Frances Brazier
S.C. members	–	Johan Helsingius
	–	Neil Todd

- **Events**

Exec in charge	–	Neil Todd & Ernst Janich
S.C. members	–	Philip Peake
	–	Johan Helsingius

- **EUnet**

Exec in charge	–	Philip Peake
Assisting	–	Frances Brazier

- **Software Distribution**

Exec in charge	–	Philip Peake
Assisting	–	Frances Brazier
S.C. members	–	Frank Kuiper

- **Relations with External Groups**

Exec in charge	–	Michel Gien
Assisting	–	Johan Helsingius
S.C. members	–	Kim Biel-Nielsen
	–	Philip Peake

- **Standards**

Exec in charge	–	Johan Helsingius
Assisting	–	Norman Hull

- **Publications**

Exec in charge	–	Frances Brazier
Assisting	–	Philip Peake
S.C. members	–	Kim Biel-Nielsen

- **Public Relations & Marketing**

Exec in charge	–	Kim Biel-Nielsen
Assisting	–	Norman Hull
S.C. members	–	Michel Gien

Michel Gien will undertake general co-ordination and responsibility for general matters.

Frances Brazier is appointed Vice Chairperson.

The priority actions resulting from the governing board decisions which the Executive Committee has set itself are to extend the scope of the EUUG, to re-examine and improve the constitution and formalise a set of by-laws, to improve co-ordination between the National Groups and internal public relations, to formalise relationships with other groups such as UniForum, USENIX and other Open Systems groups and to extend its services to target commercial needs as well as constituting to look after the technical needs of its membership.

National Groups

The European UNIX systems User Group is pleased to announce the affiliation of two National Groups – CHUUG (Switzerland) and CSHHG (Czechoslovakia). This brings to 18 covering UNIX users in France, Belgium, Denmark, Sweden, Finland, West Germany, Italy, Iceland, The Netherlands, Norway, United Kingdom, Austria, Hungary and Portugal.

Members from countries who do not have a group already formed which can affiliate to the EUUG are welcome to apply to the EUUG for direct membership. There are already many direct members from countries such as the USA and Spain. Turkey is expected to affiliate in the near future.

Events

At the recent Spring Conference and Exhibition at the Sheraton Hotel in Munich, approximately six hundred places were taken to attend the Tutorials and the main Technical Conference which covered such subjects as Operating Systems, Object Oriented Applications, Network Service, Object Oriented Programming, User Interfaces, Network File Systems, Applications and Security. The keynote speaker was Professor Richard F. Rashid who gave a presentation on the MACH operating system.

The accompanying Exhibition included stands exhibiting such well known companies as Addison-Wesley, Data General, IBM, ICL, Sequent, Siemens, SUN and Uniware.

The delegated were able to use a full e-mail service throughout the conference.

A bound 300 page book of the proceedings from the Munich Conference has been published by the EUUG and is available from the EUUG Central Office at Owles Hall, Buntingford, Herts. SG9 9PL, UK at a cost of £20.

The next EUUG Conference accompanied by a major Exhibition organised by AFUU will be held at the Acropolis Conference Centre in Nice, France fro 22nd-26th October, 1990.

Full details of this plus a call for papers for the following conference in Tromsø, Norway 20-24th May 1991 have been sent to all EUUG members.

The AFUU Report

Anne Garnery
anne@afuu.fr

AFUU

In parallel with the next EUUG Autumn Conference, the AFUU is organising at Nice, with the help of Birp-com.tec, the exhibition "NICE IS UNIX, Open System Show".It will be held from the 24th to the 26th of October at the Nice Acropolis. The entrance will be free, and all the delegates will be welcome.

Consequently, the 23th of October will be the right time for the AFUU Annual General Meeting, at Nice, from 5.00 p.m to 8.00 p.m. Il will be the occasion to present its new regional entity :AFUU Méditerranée.

The creation of regional entities has been one of the major aims of AFUU in the last three years. In order to allow all the members of the group to get and exchange information, even if they do not live near Paris, and in order to improve the development of Unix and Open Systems in all areas of France.

Three entities already exist. Each entity is headed by a G.A.R (Regional Animation Group), which develops its own promotional methods (mailings, inquiries, ..), and the activities of the area.

The purpose is not only to allow them to develop specific regional activities for the members of the area, but also to create, if possible, one or several working groups of national interest, in relation with the academic or industrial structure of the area.

AFUU/Midi-Pyrenees (South West of France) takes advantage of a machine lent by Apollo-HP and organises an efficient Fnet "T-Bone" for the Toulouse area, supplying the members with technical help and information, and coordination with the national Fnet back-bone.The GAR organises an annual conference regrouping around 150 attendees. The creation of a working group on the topic of Genie Logiciel is envisaged. Contact : Yves Thouzellier, CICT (Toulouse). Tel : +33 61 36 60 20.

AFUU/Rhone-Alpes was created on last June. On this occasion, a conference about the Portability attracted 70 attendees. They will organise the annual event "Journees Unix de Grenoble" (this year a three day conference at the end of November), they will publish a catalogue presenting the services and products, such as the computing structure of the regional members (professional and end-users), and they will initiate a working group on the topic of "Imagerie".Many other ideas: for example, they project to publish a regional letter, to organise visits of exhibitions for the regional members, to centralise the employment supply and offer, ... Contact : Serge Imbert-Bouchard (Grenoble), Tel +33 76 40 60 70.

AFUU-Méditerranée will be officially created next October, at Nice. A trainee student paid by Unisys-Nice made a survey on 150 people about Unix, Open Systems and the AFUU. More than 50 of them are interested with an affiliation and to participate to the regional entity on October.The GAR is studying the creation of a working group on the topic of Telecommunications (Videotex, EDI, ...).The meeting of October will allow the interested members to express their requirements.

Convention UNIX 91

The French Unix and Open Systems Users Group (AFUU), in cooperation with the the Bureau International des Relations Publiques (BIRP), is organising the Convention UNIX 91.

As usual, this event is structured around a series of conferences, tutorials and a commercial exhibition. The Convention will be held from the 26th to the 30th of March 1991 at the CNIT, Paris La Defense.

In order to conform to a well established tradition, the Program Committee wishes to offer to the Convention UNIX 91 visitors a wide overview of UNIX and Open Systems related issues. The range of these activities covers high level technical sessions, as well as existing or future commercial products presentations, and will also leave a fairly large part of users' experiments and analysis.

Being open and international, the Convention UNIX 91 will confirm the success it has met in the past. Three main streams of information are being set up to enable users, researchers and manufacturers to create for themselves a wide forum for exchange and dialogue:

— Technical Conferences

— Users Sessions

— Industrial Solutions

Technical Conferences

These sessions will give the opportunity to familiarise with the latest trends in research and development. The Programme Committee suggests the following subject areas:

— Distributed Systems and Cooperative Processing

— Networks: Interconnections, Interoperability and Management

— System Administration and Security

— The Rise of Open Systems

— Parallelism and New Architectures

— Graphics Applications

— Databases

— Object Oriented Applications and Interfaces

User Sessions

Today, more than ever, UNIX and the Open Systems world is ruled by the users. These sessions are intended to be a forum where these users, coming from different technical areas, cam meet and share their experiences and analysis of their hardware and software installations.

Industrial Solutions

In addition to a large exhibition where they can show and demonstrate their products; industrialists, manufactuturers and software vendors, will be able to participate in these sessions concerning existing or forthcoming products.

Method of Submission

The schedules have been settled as follows:

1st October 1990
Deadline for the receipt of communications or extended abstracts by the Conference Secretariat

15th November 1990
Notification to authors of the papers selected by the Programme Committee

10 January 1991
Deadline for the reception of the full papers by the Conference Secretariat

Submissions should include: a title, the author's name and affiliation and the target audience (Technical Conference, Users Session, Industrial Solutions).

Submissions should be sent to the following address:

A.F.U.U
Secretariat de la Convention 91
11, rue Carnot
94270 Le Kremlin Bicetre
France
Tel: +33 1 46 70 95 90
Fax: +33 1 46 58 94 20
Telex: 263887F
Net: afuuconf@inria.inria.fr

Conference Executive Chair

Christophe Binot	Sylvian Langlois
(Hewlett Packard)	(EDF - Direction des Etudes et Recherches)
+33 1 45 91 68 54	+33 1 47 65 44 02
binot@afuu.fr	sylvain@cli53an.edf.fr

Conference Proceedings Editor
Philippe Dax
(Telecom Paris)
+33 1 45 81 76 48
dax@enst.fr

"Technical Conferences" Programme Chair
Jean-Christophe Petithory
(universite Paris VIII)
+33 1 49 40 67 89 Ext 1278
jcp@uparis8.fr

"Users Session" Programme Chair
Jean-Michel Cornu
(Consultant)
+33 1 69 43 48 47

"Industrial Solutions" Programme Chair
Jean-Luc Anthoine
(IUT Belfort)
+33 84 21 01 00

The SSBA at AFUU: A Progress Report

C.Binot, P.Dax, N.Doduc, M.Gaudet
binot@afuu.fr, dax@enst.fr, ndoduc@framentec.fr

AFUU

The SSBA by the AFUU Benchmark group is 3 years old and yet very active and flourishing. Here is a progress report focusing on the 1989-1990 time frame as well as on the planned evolution toward a next release.

The SSBA: the background

The popularisation of the UNIX(tm) workstation and the recognition of UNIX(tm) as the most important operating system in the supercomputer arena as well as the feeling that it is potentially important in the conservative business-processing market, all proved to be a most appropriate ground for Benchmarking activities.

Created circa March 87, the Association Francaise des Utilisateurs d'UNIX(tm) et des Systemes Ouverts (AFUU) Benchmark group of reflects users' needs in this domain. The group wanted to provide users and consumers with a valuable and practical tool in evaluating basic characteristics of, initially, UNIX-running on personal workstations. This very clearly set goal was met quite enthusiastically by many people from different horizons, mainly users from big accounts or academia, but strangely enough, also support-staff people from French subsidiaries of American computer manufacturers for whom the group is an opportunity to understand what their mother companies are cooking and have them eat.

The work progressed the usual way thanks to the very dynamic atmosphere surrounding AFUU, and then by June '88, the very first version, 1.0, of the SSBA was fielded for tests.

With respect to EUUG, and in due time, the Benchmark group announced its creation and its activity in this Newsletter (Benchmarking in the AFUU, Vol. 7, no. 4, Winter 1987) and then made its first status report at the 1989 Spring (Bruxelles) EUUG Conference. Here, we are putting a kind of progress report concerning not only the SSBA but also the Benchmark group, and regarding the last 12 months.

La SSBA en France!

Within France, the SSBA has established itself as a mandatory item in many request-for-tender (*); many important accounts routinely request its inclusion amongst other tests or its certification (by AFUU) results, leading to a very widespread use. One vendor from a big manufacturer has even confided to us that, in 1989 alone, he has got more than 30 requests (for SSBA results) from his clients and prospects! Our statistical data for the Jan-Oct 1989 period showed that, out of 383 PD-software tapes purchased from AFUU, 89, ie 23%, are SSBA tapes. We even believe that most computer manufacturers do have a copy of the SSBA, usually and openly supplied by their French offices.

Thus the SSBA may get the credit for a strong surge of interest in the French professional media in general and magazines particularly, for this very delicate and sensitive subject. There are many points of merit to be told.

As early as the summer of 1988, Le Centre d'Experimentation du Logiciel made a survey about then existing workstations: the SSBA is used as **the** tool for this survey. Since then, at least two other similar studies have occurred and of course the SSBA has also proven its appropriateness there as well.

The display in Tribunix, AFUU's Newsletter, of the approved results began with the issue #27, Vol. 5, Jui-Aou/89 and from then on, became a regular column in Tribunix. At the beginning of 1990, we released the first special Benchmark issue of Tribunix, issue #30, Jan-Feb/90, with many contributions from people interested in this subject. Then at the conjunction of the UNIX Convention, March 1990, and the third anniversary of the Benchmark group, we built out a Dossier Benchmark Tribunix (**), that

(*) thanks to the wise and efficient lobbying of some of us ...

(**) available from AFUU secretariat.

contained selected articles from the previous issues, and most notably, all the SSBA results (about 60) collected to date. All these printed items get such a favourable welcome from many quarters that, right now, we're considering keeping them as regular items within Tribunix.

Furthermore, along with internal informational activities and beside the regular meetings (once every 6 or 7 weeks), a regular and permanent column is maintained in our Tribunix, whose name is **La Chronique des Benchmarks**, starting with issue #28, Sep-Oct/89. We keep ourselves fully informed of the happenings of different benchmarking activities (eg. when known of course, we're no newsmen and do not have their resources, although we're helped frequently by those newsmen) such as: SPEC, the Perfect Club, Utrecht's SSB ... as well as, say, InfoPC evaluation of numeric coprocessors (for i386) ...

The Benchmark group also has a rôle of a librarian regarding Benchmark software: we collect and distribute public-domain Benchmark source. Very recently, this role has been extended to articles and papers although this latter part is even more amateurish, at least at the moment, then the previous one.

The SSBA elsewhere ...

On the external front, the group and the SSBA are understandably very active too: many of its members has close contacts with corresponding entities either within or outside Europe.

The Benchmark group as is does have contacts with similar organisations like SPEC, the Performance Measurement group of UI ... and of course many of us have links with peoples working in internal Performance groups of many manufacturers.

We succeeded in having a member of the SPEC Steering Committee come and give a talk at our UNIX Convention (March '90).

The SSBA is used in product evaluation articles by IX Multiuser-Multitasking-Magazine (Hannover, Germany) and the Mai-Jun/90 issue dealt with, among many machines, the IBM RS/6000.

The SSBA is public-domain software: just ask for the EUUGD20 tape from EUUG Software Distribution. It's use is of course completely free but the publication of the results in Tribunix needs agreement from the Benchmark group.

Lastly, many copies have been sent to many educational institutions throughout Europe (Dortmund, Karlsruhe, Louvain, Liege, Valencia, ... or even to America: Worcester ...) to be used either as one of many benchmark sets to be examined within many Computer Performance courses or simply as a tool for purchase evaluation. Lately, the SSBA has also made contacts with Eastern European countries (***) and also has taken seed across the Mediterranean: Algeria, the later case within the framework of AFUU helping creating a local UNIX user group. The next release(s)

The SSBA version 1.21 has remained unchanged since Feb. 89 and in fact is **the** SSBA everybody mentioned. Having been proven in many enduring circumstances, resulting in many comments and much feed-back (****) the SSBA is now ripe for timely evolution. A Road-Map is being circulated among members for consultation and toward convergence for changes. It is most likely that before Christmas, a revised version will be released, while the other new suite of the SSBA family, although well on track, will not be ready before Spring 1991.

The actual version, 1.21, deals mainly with basic, eg. systems-level, performance will be revised and may get or include the following items: Dhrystone version 2 replacing version 1, an officially maintained version Whetstone (?), new disk throughput component(s), modifications of the SSBA's version of the Musbus ... and probably a small amount of multiprocessing measurement as well ...

The SSBA has ambitions to be a family of many suites: the version 1.x as told previously deals with generic and raw systems performance. The next suite(s) will be application-oriented, and while many domains being of immense and immediate interests, such as Real time, Graphics, Network ... are considered, we are very conservative in looking at the content of our next suite. However, it is not imprudent to state that we are quite interested, and more or less advanced in our evaluation, in items such as IOBench

(***) thanks to the *perestroika* ! Long live the *glasnost* !

(****) as well as bug reports, heated discussions and email traffic congestion.

(Wolman & Olson, Prime), Xbench (Gittinger, Siemens) ...

Another very interesting point is the analysis and use of these *raw* data obtained. Up to now, we have strived not to succumb to the temptation of giving a *single value of merit* to each machine. Recently, we have enrolled the Laboratoire de Statistiques et Probabilities de l'Universite de Toulouse to help us in processing our data. Initial results are very encouraging and we think that a new dimension of utility can and will be added to our valuable results.

The very last item in early discussion is the moving toward an EUUG-working group as suggested to and by the EC: may we wish this a real success so that the next progress report will be the first of a series of the (Euro-)Benchmark (Super-)group? **Benchmarkeurs de tous pays, unissons-nous !**

About the authors:

Christophe Binot started his commitment in UNIX when he worked at Universite de Valenciennes on MMI, and having to *choose* his first workstations, decided to invest in the hazardous ground of Workstation, Benchmarking, UNIX Culture ... all these are the names of the AFUU working groups that he chairs. When times and his AFUU vice-chairman's job permit, he also is a full-paid support engineer for HP-France. In the old times, when still at Valenciennes, Christophe many times visited his Sun 3/260 at 2 AM without proper authorisation from his young wife. (tel: +33-145916854, email: binot@afuu.fr).

Philippe Dax is an antique figure of a very old nevertheless prestigious engineering school (Sup. Telecom., the Paris establishment! please) where he used to take care of, among many things, *schoene fraulein* and *gentilles jeunes etudiantes* querying about make, yacc ... and a not-so-small set of Sun workstations. Another not-so-small activity of him is the AFUU's newsletter Tribunix that keeps growing in its scope, its thickness and the times it takes from its editor-in-chief. Philippe has put altogether, consciously, most of the SSBA script and, unintentionally, some of the yet-to-be-discovered bugs. (tel: +33-145817687; email: dax@enst.fr).

Nhuan Doduc makes his living by working in DP since his leaving theoretical physics, and makes his pleasure by benchmarking a not-too-unknown FORTRAN program since very long time ago. Nhuan is a strong driving force within many AFUU's groups (Calculateurs Scientifiques ...), spends much of his time in contributing to Philippe's Tribunix, participates actively in the G.U.S, (aka. the French S.U.G.) and the Workstation group of the CUBE, (the French Bull user group) ... and, for what is left of his time, also officially works for Framentec and Cognitech, two expert-system software houses. Nhuan is the *keeper* of the benchmark library of the group. (tel: +33-147964633; email: ndoduc@framentec.fr).

Michel Gaudet stills work in Laboratoire de Biorheologie Hydrodynamique PhysicoChimique of Universite Paris VII, where he reigns supreme on many PCs openly connected (eg. cables disconnected most of the times) to a 4Mb Sun 3/140. Beyond UNIX, Michel's many expertises include the PC hardware, Ms/Dos, the nonfunctioning of the SSBA's Musbus, the I/O rates from the Saxer as well as from many other I/O benchs, but then he is still hesitant on deciding to move his Sun to OS 4.1, that is with as many as 4 (four) Mb of Ram. While waiting for 4.1 to arrive, he makes lots of travels to take care of most of the SSBA results. (tel: +33-143362525 ext 4333).

Request for Test

The AFUU has decided to make an open way in which the UNIX users' community can access the computer performance information printed in Tribunix.

With this in mind the AFUU has offere those producing and distributing UNIX based systems to publish in Tribunix, under their own name and with the AFUU certificate of testing, the output obtained from running the SSBA 1.21F, with the aim of informing the membership of the AFUU. A request to print this in Tribunix must be supported by a certificate of having run the benchmark in the presence of a person approved by AFUU for this purpose (Nhuan Doduc, Philippe Dax, Michel Gaudet). Also needed is the location of the test, electronic evidence, and the output of the test.

Dominique Maisonneuve, Christophe Binot, Nhuan Doduc, Philippe Dax.

An example of test output follows:

SYNTHESE DES RESULTATS DE LA SSBA 1.21F (27/02/89)	
Identification : BULL DPX/2 340 Moto 68030/68882 B.O.S. 1.1 33 Mhz 16 Mbytes 2*300 Mbytes	
ENVIRONEMENT	
Type of Unix	SVR3
Command C	cc -FO -DTERMIO -DSysV
Command F	f77 -FO -DTERMIO -DSysV
Name of site	B.O.S.
Name of login of benchmarker	root
Value of HZ	100
Number of processors before start	6
Number of processors staying for the execution	251
Available virtual memory space	0 Mb
RESULTS	
Date of start	Thu May 10 11:56:03 EET 1990
Mips/Joy	5.63636
Dhrystone (without registers,without optimisation)	11483 Dhry/s
Dhrystone (with registers,without optimisation)	11483 Dhry/s
Dhrystone (without registers,with optimisation)	12860 Dhry/s
Dhrystone (with registers,with optimisation)	12879 Dhry/s
Whetstone (single precision)	2867 KWhet/s
Whetstone (double precision)	2637 KWhet/s
Linpack (single precision, rolled)	330 Kflops
Linpack (single precision, unrolled)	341 Kflops
Linpack (double precision, rolled)	274 Kflops
Linpack (double precision, unrolled)	280 Kflops
Utah (execution time)	user=12.22s, sys=2.8s, u+s=15.02s
Outils (execution time)	user=52.84s, sys=7.9s, u+s=60.74s
Byte (execution time)	user=2.28s, sys=11.96s, u+s=14.24s
Saxer (thoughput W/S disc)	548.24 Kb/s
Testc (execution time)	user=66.30s, sys=1.72s, u+s=68.02s
Doduc (double precision)	iterations=5491, temps=946.18s, ratio=51
Bsd memory (execution time)	user=82.74s, sys=1.72s, u+s=84.46s
Bsd syscalls (execution time)	user=0.64s, sys=16.98s, u+s=17.62s
Bsd pipes (execution time)	user=0.84s, sys=43.12s, u+s=43.96s
Bsd forks/execs (execution time)	user=3.2s, sys=214.74s, u+s=217.94s
Musbus (thoughput W/S disc for 62 blocks)	write=1033.3 Kb/s read=1550.0 Kb/s copy=620.0 Kb/s
Musbus (thoughput W/S disc for 125 blocks)	write=446.4 Kb/s read=2083.3 Kb/s copy=420.2 Kb/s
Musbus (thoughput W/S disc for 250 blocks)	write=740.7 Kb/s read=1785.7 Kb/s copy=554.5 Kb/s
Musbus (thoughput W/S disc for 500 blocks)	write=979.6 Kb/s read=1785.7 Kb/s copy=306.1 Kb/s
Musbus (thoughput W/S disc for 1000 blocks)	write=1173.7 Kb/s read=503.4 Kb/s copy=139.8 Kb/s
Musbus real time	794.00s
Musbus cpu time + system	665.37s
Average size of an executable	39.8967 Kb
Total Time	5977s
Number of cycles	78
Average number of users	1
Average number of processors	17
Date of end	Thu May 10 13:35:40 EET 1990

COMMENTAIRES

Passage chez : BULL MTS - 1 rue of Provence - Parc Surieux BP208 - 38432 Echirolles Ceofx

Marie-Hlne Marc et Michel Gauoft

PRELIMINARY ANNOUNCEMENT CALL FOR PARTICIPATION INTERNATIONAL WORKSHOP ON UNIX-BASED SOFTWARE DEVELOPMENT ENVIRONMENTS

Hotel Grand Kempinski, Dallas, Texas, USA
16-18 January 1991

Co-sponsored by:
USENIX Association (USA)
SIGMA Project (Japan)

Many software development environments have been described, built, or used which are intended to operate atop the UNIX system. The goal of this workshop is to share information on what these systems look like, what problems were solved by using UNIX and what problems were caused by it. We expect strong representation from the Japanese SIGMA workstation project, which defines a national software engineer ing environment that uses UNIX as its base, as well as American and European academic and industrial organisation.

Participants will be selected by a program committee on the basis of submitted position papers. Attendance will be limited to 75 to encourage discussion. Meetings will include descriptions of important systems and presentations on particular technical points involving implementation and usage. Significant time will be set aside for panels and informal discussions of such topics.

Position papers will be evaluated by a program committee including researchers and practitioners from Europe, Japan, and the United States. Please send a 1-4 page position paper by August 1, 1990 to one of the co-chairs:

Noboru Akima	Stuart Feldman
Sigma Project	Bellcore
5th Akihabara Sanwa Bank Building	445 South Street
3-16-8 Soto-Kanda, Chiyoda-ku	Morristown, NJ 07962-1910
Tokyo, Japan 101	USA

Electronic versions may be mailed to sdeconf@bellcore.com

WORKSHOP CONTENT

Relevant topics that might be addressed in the position paper include:

- Description of a significant system (by a designer or builder)
- Experience with using such a system
  - — novel tools or facilities offered by such a system
  - — evaluations of usage
  - — positive and negative experiences

- Experience with building such a system
  - architectural considerations for a UNIX-based Software Development Environment
  - advantages resulting from basing the system on UNIX
  - problems (and solutions) encountered in designing and implementing such a system (e.g. file and database systems, networking and cooperation, scheduling and resource usage)

WORKSHOP FORMAT

The structure of the workshop will be decided after participants are selected. A likely agenda is:

Wed		Descriptions of SIGMA project
Wed		Descriptions of other large systems
Wed		Discussion of technical difficulties
Wed	eve	Group reception and Birds of a Feather sessions
Thu	morn	Panels on topics arising on Wednesday
Thu	aft	Subgroups
Thu	eve	Birds of a Feather
Fri	morn	Subgroup reports
Fri	aft	Panels, debate, wrapup

UKUUG Report

Mick Farmer
mick@cs.bbk.ac.uk

Birkbeck College

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

Start Bit

This report is shorter than normal because I'm catching up with the work left to one side during our recent London conference.

UNIX The Legend Evolves

We held a very successful conference last July at the Royal Lancaster Hotel in London (England). Over 440 people attended the conference which naturally attracted the interest of the computing press. The likes of Brian Kernighan, Rob Pike, Dennis Ritchie and Ken Thompson were forever fending off questions like *"How did you start to ..."*. Alas, one journal managed to get the event wrong and called it the annual meeting of the European UNIX User Group!

The conference is described in more detail elsewhere in this issue (two of the Plan 9 papers are being reprinted and there's the usual conference report) so I'll conclude this piece by mentioning the time-honoured competition quiz. Rob Pike invited us to invent a suitable insult for the X Window system. From a mountain of entries the judges (Rob, Dennis, Ken) chose the following:

1. X is a terminal illness.
2. X a trip down "memory" lane enabling the user to experience the performance of a decade ago, but with no discernable advantages.
3. X is just bad NeWS.
4. Light travels half as fast through an X-window.

Our thanks to Addison-Wesley, Prentice-Hall and Wiley for donating books for the prizes.

Conference Proceedings

Copies of our London conference proceedings (undoubtably a collector's item for the future) are available from the UKUUG Secretariat (address on the back cover of this Newsletter) at £50.00 each.

Yet More Events

- Our Winter '90 Technical Meeting will take place at Queen's College, Cambridge (England) on 17-19 December, 1990. As with all our winter meetings, this one will have a strong networking flavour.
- Our Summer '91 meeting will be held in Liverpool (England) in June or July 1991.
- Our Winter '91 meeting will take place in Edinburgh (Scotland) in December 1991.

Please note that EUUG members and members of other national groups are always welcome to attend our events.

Workshop Videos

We have produced two video programmes on relevant material to those working in the UNIX community. Both of these are the result of successful one-day workshops organised by the UKUUG.

- UNIX Security

  A three hour video discussing the following topics:

  — The HACKMAN project; System V/MLS; An analysis of the Internet worm.

  — The Sun Yellow Pages system; Secure RPC; Some myths and facts about UNIX security.

  — And more...

- UNIX System Administration

  A four hour video discussing the following topics:

  — POSIX developments; System management; Managing X.400 mail systems.

  — Project Athena; System administration in a heterogeneous environment.

  — And more...

Each video costs £60.00 (plus VAT in the UK) and can be ordered from the UKUUG Secretariat, by e-mail from ukuug-videos@ukc.ac.uk, or directly from:

Birkbeck College Video Services
Department of Computer Science
Birkbeck College
Malet Street
London WC1E 7HX

+44 71 631 6351

London UNIX User Group (LUUG)

This lively group continues to meet on the last Thursday of every month (except December). Contact Andrew Findlay (Andrew.Findlay@brunel.ac.uk) for details.

Midlands UNIX Group (MUG)

A number of individuals and organisations have shown an interest in getting this off the ground. Contact Kevin Hopkins (mug-request@uk.ac.nott.cs) or David Osbourne (cczdao@uk.ac.nott.clan) for details.

We wish them, and any other embryonic group, all the best. The UKUUG is always willing to assist the formation of local UNIX groups throughout the country.

FaceServer Project

This service is now in full swing and we intend taking it to most major conferences in the future. Note that the name has changed ever so slightly due to a registered trademark problem.

This project is being supported by Acorn Computers Ltd. of Cambridge (England) as part of their on-going commitment to UNIX.

Software Distribution Service

We currently have over 600 megabytes of (compressed) source archives on-line for UKUUG members to access. The distribution service is offering the same service to EUUG members (through their national groups) at the same price. See the article by Lee McLoughlin and Stuart McRobert elsewhere in this issue for further details. E-mail enquiries to ukuug-software@doc.ic.ac.uk.

Stop Bit

For the serious bird watchers (twitchers :-) among us there's always EuroBirdNet for the latest information. Now where can I see Curlew Sandpipers together with Savi's Warblers if I get there fast enough ...

X Server for Sun®

SPARC®, Sun/3, Sun/386i

- Sunview® concurrency
- GX® graphics support
- Easy installation
- Simple configuration
- Comprehensive support

OSF/Motif® for Sun

SPARC, Sun/3, Sun/386i

- Motif Window Manager
- Shared libraries
- Full development system
- Easy installation
- Simple configuration

Motif Programming Course

- Developed by OSF
- Hands-on Workshop
- Build real applications
- Topics include
  - Introduction to X
  - Using widgets
  - Prototyping with UIL

Unipalm XTech
145 St. Neots Road, Hardwick, Cambridge CB3 7QJ
England
Tel : +44 954 211862 Fax : +44 954 211244
All trademarks acknowledged

UKUUG Software Distribution

Lee McLoughlin
Stuart McRobert
ukuug-soft@doc.ic.ac.uk

Department of Computing
Imperial College
180 Queen's Gate
London
England
SW7 2BZ

The UKUUG Software Distribution Service

The UKUUG has over 600 megabytes of (compressed) source archives on-line for UKUUG members to access. This archive is intended to allow UKUUG members to easily access major distributions of freely distributable software. It is also available to EUUG members.

One of the aims of this archive is to act as a place where you can obtain all those interesting packages you read about on the Usenet news but cannot get because they are only available on remote networks.

Contents

Here is a heavily edited list of the highlights. Where possible these are the latest versions available and are complete distributions. Ordering information is in the form *package-format-tapes*.

X.V11R4

The X Window System, Version 11, Release 4. This is the original release tape with copies of the official patches. (See also *X.UNOFF* below).

Ordering:

- **X.V11R4** 1 tape
- **X.V11R4** 2 tapes
- **X.V11R4** 1 tape

X.UNOFF

Unofficial additions and upgrades to the X.V11R4 system. This is a copy of the Usenet newsgroup `comp.sources.x` and the user contributed areas copied from various X.V11R4 archive sites around the world.

Ordering:

- **X.UNOFF** 1 tape
- **X.UNOFF** 2 tapes
- **X.UNOFF** 2 tapes

Comp.sources

Complete copies of the UNIX related source Usenet news groups: `comp.sources.unix`, `comp.sources.x`, `comp.sources.sun` and `comp.sources.misc`.

Ordering:

- **COMPSRCS** 1 tape
- **COMPSRCS** 2 tapes
- **COMPSRCS** 2 tapes

TeX

TeX typesetting system, including sources, styles, formats, previewers, ...

Ordering:

- **TEX** 1 tape
- **TEX** 1 tape
- **TEX** 1 tape

GNU

All the latest releases from the Free Software Foundation. Including GNU Emacs, the GNU C compiler, the GNU C++ compiler, and clones of many other programs.

Ordering:

- **GNU** 1 tape
- **GNU** 2 tapes
- **GNU** 1 tape

UNIX

+ Athena + Sun

A grab bag of general UNIX kernel related utilities, the Athena software, for managing networks of workstations, and Sun specific upgrades.

Ordering:

- **UAS** 1 tape
- **UAS** 2 tapes
- **UAS** 1 tape

Comp.os.minix

A pruned copy of the Usenet news group about the Minix operating system.

Ordering:

- **MINIX** 1 tape
- **MINIX** 1 tape
- **MINIX** 1 tape

Comms

Communications, mail, and news software. This includes ISO-DE, SNMP, name servers, slip, gated, tcpdump, niftp, spad, tunnel, kermit, pcomm, xcomm.

Ordering:

- **COMMS** 1 tape
- **COMMS** 2 tapes
- **COMMS** 2 tapes

Languages

Languages including sbprolog, KCL, various dialects of ML, perl, various lisp's, leif, icon, modula-3, ...

Ordering:

- **LANG** 1 tape
- **LANG** 2 tapes
- **LANG** 1 tape

Availability

The distribution is available both via tape and electronically. Electronically the software is free. However there is a charge for tape distributions. This charge is kept as low as possible.

General Enquires

General enquires can be handled by email to: ukuug-software@doc.ic.ac.uk

Electronic Access

Interactive
Outside normal working hours (8.30 to 17.30) a guest login account is available for those who simply wish to wander around the archive. It is accessible at Imperial College via a terminal concentrator accessible over the Janet X.25 network with an address of:

```
000005101000    cudf=DOC.G
```

When talking to a pad the call would look like:

```
call a000005101000 d=DOC.G
```

or perhaps

```
call .000005101000*DDOC.G
```

It is not yet registered with in NRS database so an absolute address has to be given.

Once connected to the concentrator you should have a conversation like:

```
Enter User Group (or type help): sources
Enter Service (or help): sources
Calling
CONNECTED

login: sources
Welcome to the UKUUG Software Distribution
    Service at uk.ac.ic.doc.src
ic.doc.src$ help
```

Once connected you are in a restricted environment. Use the `help` command for up to date details of what commands are available.

File Transfer
All material in the archive is available via Janet (and therefore PSS) using Niftp. On your own system the Niftp command will be called, `fcp`, `cpf` or `hhcp`. Please do not confuse this with the common command `ftp` which normally operates only over local ethernets.

The basic details required for the niftp are:

```
Host: uk.ac.ic.doc.src
Name: guest
Pass: your email addr
```

All filenames are prefixed by `<PUB>`. The majority of files on the archive are binary. In fact most files are compressed in order to save space, this is denoted by the `.Z` suffix. So all files ending in this suffix are binary, but so are many others, in particular the X.V11R4 files `tape-?.??`. This is usually important to niftp systems. If an attempted binary transfer is rejected with a message indicating that the file was really text, retry specifying the text option to your niftp program.

To find what files are available first pull back the file `<PUB>ls-lR.Z` which is a compressed recursive directory listing of the entire archive. This is updated nightly and is currently about a quarter of a megabyte in its compressed form!

This file can be pulled back by using a command like:

```
csh% fcp -b "<PUB>ls-lR.Z@\
    uk.ac.ic.doc.src" ls-lR.Z
User: guest
Password: lmjm@doc.ic.ac.uk
csh%
```

Note that since the source filename contains the < and > characters this argument has to be quoted.

Note also that a line has been split at the \ to fit in the column.

Once retrieved this file can be viewed with a command like:

```
csh% uncompress < ls-lR.Z | more
```

Getting the Distribution via Tape.

Currently there are several restrictions on access via tape. Firstly to non-UKUUG/EUUG members there is a 30% surcharge (this can be considered yet another good reason to join the UKUUG). Payment, if by cheque, must be in Pounds Sterling on a UK bank. Lastly the tapes are in `tar` format on one of the following kinds and densities of tapes:

```
1600 bpi half inch
6250 bpi half inch
QIC24
```

This corresponds to the *format* field of the ordering information.

The total cost is a multiple of the total number of tapes required. Currently this is £50.00 per tape.

If the order is to be sent to a UK address we have to charge 15% VAT. If the order is to be sent outside of the UK we do not have to charge VAT but the postal charges are 15% more. So either way add 15% to the cost of the tapes.

If the payment is enclosed with the order then we hope to have a turn around time in the order of 7 days. In which case send to the address:

```
UKUUG Software Distribution Service.
c/o Lee McLoughlin,
Department of Computing,
Imperial College,
180 Queens Gate,
London.
SW7 2BZ
UK
```

An order form follows on the next page.

The next page may be photocopied for use.

If the payment is not included with the order, for example if using a purchase order, then please submit the order to the UKUUG secretariat so they can clear the order.

```
UKUUG Software Distribution Service.
c/o Bill Barrett,
UKUUG,
Owles Hall,
Buntingford,
Hertfordshire.
SG9 9PL
UK
```

Product	Format	# Tapes per Set		Number of Sets ordered		Total # of Tapes	Notes
X.V11R4	6250	1	x	0	=	0	Example
			x		=		
			x		=		
			x		=		
			x		=		
			x		=		
			x		=		
			x		=		
			x		=		
			x		=		
			x		=		

£50.00 x Grand total number of tapes	Sum this Column	£
30% surcharge if NOT a UKUUG/EUUG member		£
UK address add 15% VAT, non-UK address add 15% p&p		£
Grand total including VAT or p&p		£

I declare to indemnify the UKUUG for any liability concerning the rights to this software, and I accept that UKUUG, and its agents, takes no responsibilities concerning the contents and proper function of the software.

Name ____________________

Signed ____________________

Email Address ____________________

Phone Number ____________________

Delivery Address ____________________

EUnet Column

Miss Anke Goos
ag@unido.uucp

University of Dortmund

Originally a journalist and student of journalism with a focus on "new media" at the University of Dortmund, she got a hooked onto Unix by a job for User Information in the German EUnet backbone. She got a dubious reputation for work on the European E-Mail Directory, the EUnet glossy, German EUnet documentation, editing the Newsletter of the German Unix User Group and annoying her technical colleagues in the EUnet backbone with reminds about "what the users want". Besides, she is writing articles for German (Unix) magazines, if she can't defend it and tries to pursue her studies, currently with a research paper about - "surprise, surprise" - European computer networks.

The following is a short excerpt of several important points of EUnet activity over the last few months. This report was produced with the help of Yves Devillers, Ted Lindgreen and Frances Brazier.

Backbones Changes

The Belgian backbone changed from Philips to the University of Leuven in March and the Norwegien backbone has been transferred from Norsk Data to the University of Oslo. No EUnet backbones are now based at commercial organisations anymore.

In April the Swiss switched to SWITCH, now situated at the Eidgenössische Technische Hochschule Zürich. SWITCH is an organisation which provides telecommunication services, such as a leased line network for the Swiss Academic community. The former backbone CERN will retain a supernational status for the CERN network.

Two new backbones are starting up: One in Hungary at the Institute of Research and Computer Automation in Budapest. Another one in Czechoslovakia at the University of Technology in Bratislavia. Their candidacy has been agreed on in the backbone meeting.

Estonia had asked for a link to EUnet via Finland early last year. This had caused wide discussions in the EUnet backbone mailing list in Summer 1989 how to handle requests from regional backbones without the full support of the National Unix User Group (NaLUUG), in this case the Russian UUG. The common consensus was that EUnet doesn't want to be exploited for political purposes.

InterEUnet

You may already know that InterEUnet is a pan-European TCP/IP service like the American Internet and the Scandinavian NORDUnet. InterEUnet provides IP-access to all these networks. The extension of the already existing InterEUnet services throughout all European countries has been studied by the backbone community since spring 1989. This project will cause an increase of the EUnet budget by more than an order of magnitude. A study by Peter Collinson concluded that a more structured approach to EUnet is required. Based on these results the EUUG has decided to support the InterEUnet project with 65,000 ECUs for initial activities such as market research and to solicit applications for professional management.

International Character Sets

Keld Simonsen made a proposal to introduce support for international character sets using 8-bit data. The backbone meeting has advised all backbones to adopt this and Keld was asked to report on the experiences.

Clarinet

EUnet will carry special closed newsgroups such as the commercial news service Clarinet. In principle, EUnet will charge the data provider for this service, depending on the amount of data transported. For the time being, Clarinet will not be charged as the amount of data transported is negligible.

Breukelen Backbone Workshop

During the EUnet backbone workshop in Breukelen in July several working groups have turned out future plans, which might be of interest for the EUnet user:

First of all, the second version of the European E-Mail Directory should be in Yugoslavia at the time of your reading this. This is at a substantially lower price than last year and a higher number of 1500. This might give you an opportunity now to acquire a fresh new update of all the European network sites of EUnet and EARN.

The plan of a EUnet-wide archive service, distributed between the backbones and other interested secondary feeders should be working from October. This in responsibility of one person, financed by EUnet for six months initially. The service should be open to both EUnet and EARN users.

Due to the increasing usage of lines by IP-traffic, higher changes for the leased lines are to be expected. Examples may be seen between CWI and CERN as well as the German backbone Unido and CWI. You, as a user, are not supposed to see too much of these network activities :-), though.

Contracts with the Scandinavian Nordunet about sharing the CWI-US link in exchange for the backup line Nordunet-US are under way, formalising what has already been common practice in network cooperation for some time.

PACCOM, which links Japan, Australia, New Zealand and Korea is looking for a leased line to Europe. EUnet might participate in a share of 10,000 ECU per year, cooperating with HEPNET, Nordunet and the national academic networks.

*** Telefax and Telex for UNIX ***

Send and receive, Telefax and Telex directly from your UNIX machine. Shortcodes, transmission at a later time, userfriendly interface, etc. etc.

Contact Tonny Andersen, at ACS on international telephone +45 30 45 48 16 or EUnet tonny@acs.dk

USL Column

Chris Schoettle

UNIX System Laboratories Europe

For further information on this column please contact Gill Mogg on *gill@uel.uucp*. Gill is Market Communications Manager at USLE. The guest writer this issue is Chris Schoettle.

Chris Schoettle is a Senior Consultant with UNIX System Laboratories based in London, England. Mr. Schoettle has BA and MSc degrees in Computer Science and has worked extensively in the international UNIX systems market. He has presented the technical contents of UNIX System V Release 4.0 and of the OPEN LOOK Graphical User Interface thoughout Europe, including the EUUG, UNIForum, and USL's Software Developer Conferences. He represents USL on the X/Open User Interface Working Group.

OPEN LOOK - A Consistent Approach to a GUI Architecture

Much has been written, discussed and even hyped on Graphical User Interfaces (GUIs) to the UNIX Operating System. It is becoming increasingly more difficult to grasp the underlying principles on which the products are based. Here, we will address the rationale that UNIX System Laboratories (USL) has followed in developing and licensing the software that implements the OPEN LOOK® GUI.

A GUI comprises three functional areas:

- a platform
- a programming interface
- a user perspective

Before attempting to define the above terms, it is worth drawing an analogy (albeit over-simplified). Consider a GUI as an automobile. The platform is the entire interworkings provided in the base vehicle. There might also be an optional extra such as an on-board computer. The programming interface of the computer can be thought of as that part which interoperates with the interworkings of the automobile.

The user perspective of the computer is what the driver sees on the face of the computer and how the driver operates it. In general terms, the programming interface of the automobile is how the base car and all the optional extras interoperate. The user perspective is everything that the driver sees and operates in the car, including the base and optional extras.

Now let's replace the world of automobiles with that of GUIs. The platform is the software which manages the window system (the base car), including the display and input device. Other software, such as a toolkit, can be provided on top of the platform (an optional extra), and is usually dependent on the platform on which it is implemented.

The programming interface is the syntax and semantics defined for use by applications interfacing with a platform and possibly a toolkit. A toolkit can provide functionality in addition to that provided by the platform; this can include components such as buttons, scrollbars and menus. For example, a toolkit may provide software to implement a scrollbar in an application. The programming interface for this scrollbar would most likely be a combination of the semantics defined by the platform and the syntax defined for the scrollbar in the toolkit.

The user perspective is what the end user will see and operate with on the terminal, irrespective of the underlying software; this is widely termed as the Look and Feel. In our example of the scrollbar, the user perspective of the scrollbar is what it actually looks like when seen on the terminal and how the user uses the mouse to operate the scrollbar within an application.

In early GUIs, such as SunView* or the Macintosh*, the GUI is inherently part of the platform, operating system, and even hardware; these are kernel-based window systems. As the industry has moved towards the Open Systems marketplace, client-server based window systems have evolved as platforms. These provide an environment for networking and portability, across different operating systems and hardware, and as a base for different GUI architectures. Two client-server based window systems of note are the X Window System* and NeWS*, the network extensible Window System.

The X Window System, or X, is a defacto (and now X/Open official) standard, developed by the Massachusetts Institute of Technology. NeWS is a PostScript* based architecture developed by Sun Microsystems. The programming interface, unlike the user perspective, of a GUI for a platform can be dependent on that platform, in this case X or NeWS. Specifically with the OPEN LOOK GUI, while there is one user perspective, there are three different programming interfaces, different by both design and objective.

The OPEN LOOK X Toolkit (code named Xt+) is a widget set built at the Xt Intrinsics layer of X (this is just X terminology for saying that it is a set of components built on the top layer of the X platform). The programming interface for the OPEN LOOK X Toolkit is a combination of the syntax provided in the widgets of the OPEN LOOK X Toolkit and the semantics of X. It is the one which is most commonly compared to the OSF/Motif* Toolkit.

The OPEN LOOK XView* Toolkit has a programming interface very close to that of SunView, a kernel-based window system. It has been designed with the purpose of migrating existing SunView applications to X (and the Open Systems marketplace). To maintain the SunView programming interface, the OPEN LOOK XView Toolkit has been developed at the Xlib layer of X, a lower layer than the Xt Intrinsics.

The OPEN LOOK tNt (the NeWS Toolkit) has been developed in PostScript to provide a programming interface for software developers wanting to develop OPEN LOOK applications on NeWS.

Thus, for a software developer there are three different programming interfaces for the OPEN LOOK GUI. For the end user though, there is just one user perspective, OPEN LOOK. The Look and Feel for OPEN LOOK is strictly defined in a set of published books known as the OPEN LOOK Specification and Style Guides. This Look and Feel has gone through an extensive industry review process. It is published so that software developers have the opportunity of developing new toolkits which implement the Look and Feel of OPEN LOOK. Thus, there is the possibility of having additional programming interfaces (in addition to the three already mentioned) that have been designed to meet a market requirement for implementing the Look and Feel of OPEN LOOK. One such additional interface is the OI Library from Solbourne which has been designed for the C++ market and is discussed further below.

What a user reads in the OPEN LOOK Specification is exactly what the user will see on the terminal. The user will not be able to distinguish (and will probably not care) which platform and OPEN LOOK toolkit programming interface was used to develop the application. For the user it is all the same, OPEN LOOK.

Let's return now to OPEN LOOK product issues. OPEN LOOK GUI Release 2.0 is the most recent release of the OPEN LOOK X Toolkit. It builds on the capabilities provided in OPEN LOOK GUI Release 1.0, first released in March 1989. Without changing the programming interface, it provides the final Look and Feel defined in the OPEN LOOK Specification. It has reduced memory requirements and performance improvements with gadgets and flat widgets (more object-oriented by not forcing each component to be window). It is based on X Window System Version 11 Release 3 (X11R3); a later release will be based on X11R4. It has an end user environment which includes the applications Window Manager, Workspace Manager, Terminal Emulator, Pixmap Editor, Print Screen, and File Manager. The File Manager provides a graphical user interface to the UNIX System V file system and is the beginning of a desktop metaphor.

The X Window System has been productised by USL into XWIN. XWIN Release 3.0 is based on X11R3 (a later release will be based on X11R4) and contains many improvements over the public domain X from MIT. XWIN Release 3.0 has reduced memory requirements, many bug fixes, is implemented on STREAMS, uses shared libraries, provides a colour Terminal Emulator, and has much improved performance.

NeWS has been released by USL in a merged window system architecture with X, called X11/NeWS*. Recognising that both X and NeWS are client-server based window systems with commonalities at the server level, X11/NeWS contains a single merged server for both X and NeWS. It can be thought of as a single room containing such items as a common window tree and event queue, but with two doors for entering, one for the X Protocol and one for the NeWS PostScript protocol.

The graphics products we have discussed are all available now. All are included in either the source code of UNIX System V Release 4, as a separate bundled graphics product called Graphics Services Version 2, or as separate individual products.

The OPEN LOOK software has a consistent approach to a GUI architecture. This is pictorially represented in Figure 1. Here we have a platform of UNIX System V Release 4 with XWIN and X11/NeWS. At the level of the programming interface, there are three OPEN LOOK toolkits. In developing an application, a software developer chooses the OPEN LOOK toolkit with the most suitable programming interface.

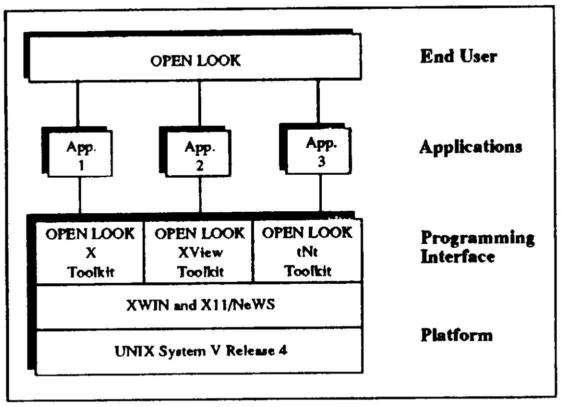


Figure 1: OPEN LOOK GUI Architecture

Application 1 is most likely a new application built at the Xt Intrinsics layer of X and so uses the OPEN LOOK X Toolkit. Application 2 has probably been recently converted from SunView and so uses the OPEN LOOK XView Toolkit. Application 3 could be a PostScript application on NeWS and so uses the OPEN LOOK tNt Toolkit. A different programming interface has been designed to meet the design objectives for each of the applications. From the end user perspective it is all the same, OPEN LOOK, just as it has been defined in the OPEN LOOK Specification. Two platforms, three toolkits, one Look and Feel - this is the consistent approach of OPEN LOOK.

Before leaving, it is worth looking at the future of OPEN LOOK. One of the most commonly asked questions is in comparison to the Motif Toolkit as to why USL doesnUt cancel OPEN LOOK and use Motif. This is a question we asked both ourselves and our customers in Autumn 1989. The internal USL answer is clear; we feel OPEN LOOK is a technically superior product with much to offer the Open Systems marketplace. We could go through a list of reasons here, but it would soon become clear that there are not many major differences to highlight. There are many smaller differences, but even these are becoming less obvious in ensuing releases of OPEN LOOK and Motif.

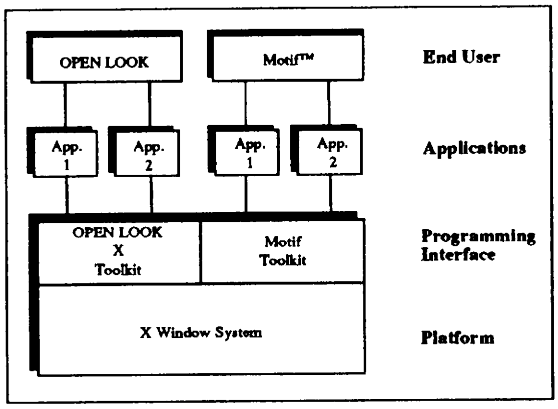


Figure 2

There are two primary differences that are worth looking at between OPEN LOOK and Motif. One is the different Look and Feel, but this is becoming less of an issue as the market is recognising that there are different market areas for different Look and Feels and that this is even desirable. The other difference is the programming interface. The three OPEN LOOK toolkits have three different programming interfaces by both design and objective. The OPEN LOOK X Toolkit and the Motif Toolkit are both widget sets at the Xt Intrinsics layer of X, and there is no need for them to have different programming interfaces. The OPEN LOOK X Toolkit and the Motif Toolkit do not have different programming interfaces by design and objective, they have different programming interfaces by accident.

The result is shown in Figure 2, where Applications 1 and 2 have been developed with the OPEN LOOK X Toolkit to use the OPEN LOOK Look and Feel. If there is a need for Applications 1 and 2 to use the Motif Look and Feel, they have to be ported to use the Motif programming interface. This is unacceptable to software developers and was agreed as the most significant problem facing the X/Open Desktop Computing Workgroup.

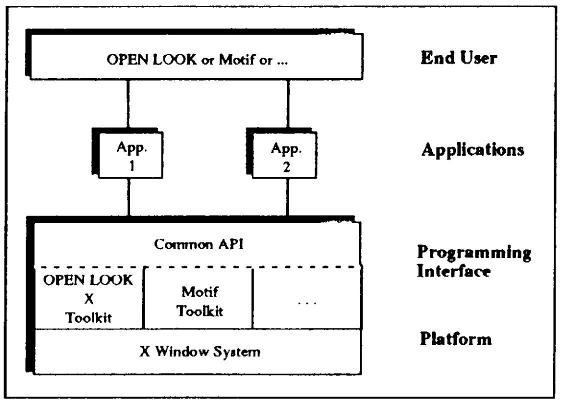


Figure 3

To help resolve this situation, the IEEE P1201 standards group is working on defining a Common API (Application Programming Interface) to OPEN LOOK and Motif. Pictorially, the result should be similar to Figure 3, where Application 1 and Application 2 have been written to a single Common API and can effectively toggle between the OPEN LOOK and Motif Look and Feels, without having to be ported. USL and UNIX International fully support a Common API and are helping to define a standard through the IEEE P1201 forum. We would welcome support from all interested parties.

While a Common API standard is being defined, several companies have already made the strategic decision of developing their own single programming interfaces incorporating both OPEN LOOK and Motif Look and Feels. One such company is Solbourne which has developed the OI Library, a single C++ interface which provides the Look and Feel of both OPEN LOOK and Motif. We consider software such as the OI Library from Solbourne as proof of existence that a Common API is definitely feasible.

Going back to the original question: why is USL continuing with the OPEN LOOK strategy? We have already said that from an internal USL perspective we feel it is a better product, but what do our customers think? After all, if our customers do not want a product, then it takes little market awareness to realise that we should not sell it. In Autumn 1989 we asked our customers what they thought about OPEN LOOK.

• Shipping Product:	
AT&T CSB	Quest Systems Corp. (S/ware)
AT&T USL (Software)	Solbourne Computer, Inc.
Harris Corporation	Sun Microsystems, Inc.
Quantum Software Systems Ltd	
• Commitment:	
Altos India	Labtam
Amdahl	Motorola
ARIX	Olivetti
Commodore	Pyramid
Fujitsu	Sequoia
HCL Ltd.	SONY
ICL	Tolerant Software (Software)
Intel	TOSHIBA
INTERACTIVE (Software)	Wipro
KonsultHusetData (Software)	

Figure 4: OPEN LOOK GUI OEM Commitment

The answer has been a resounding response in favour of OPEN LOOK. USL is keeping OPEN LOOK because our customers asked us to. In Figure 4 is a list of OEMs who have committed to ship OPEN LOOK. In Figure 5 is a list of ISVs who have committed to develop OPEN LOOK applications.

With the future of OPEN LOOK decided, USL is now working on that future. OPEN LOOK will follow the path determined in UNIX International and in standards organisations such as X/Open and IEEE P1201.

USL is now developing internationalisation capabilities in OPEN LOOK. We are doing this in conjunction with UNIX International companies, such as Fujitsu. A prototype of a future OPEN LOOK internationalisation product is now being made available to UNIX International companies.

It is straightforward to realise that Graphical User Interfaces will be an imprtant part of the future of the UNIX Operating System. OPEN LOOK is very much a part of that future with UNIX System V.

• Existing Applications:

Bristol Group (Fax Software)	Ingres (DEMS Interface)
Elan (Text Processing)	VersaSoft (dBASE Workalike)
EXOC (Prototyping Tools)	Visual Edge (UIMS)
Ficor (Business Graphics)	

• Developing Product:

Advanced Software Automation	(CASE)
ADAPTI	(Financial)
Answer Computer	(CASE)
Arbor Text	(E-pubs)
Ashton-Tate	(Database)
Auto-Desk	(Arch/Engr)
BBN	(Data Analysis)
Bristol Group	(Visualisation)
Conceptual Structures	(E-pubs)
Control Data	(Education)
Convergent Solution	(Database)
Crosswind Systems	(Office Auto)
Cullinet/CA	(Database)
Digital Solutions Inc.	(Financial Svsc)
Entropic speech	(Signal Processing)
Expert Object Corp	(CASE)
Gateway Design	(Electronic Design)
Georgetown University	(Medical)
Graphic Software Systems Inc.	
H.T. Graphics	(Financial Services)
IDE	(CASE)
Information Builders	(Focus - DBMS)
Informix	(Database)
IXI	(Desktop)
Lanson-Davis	
Lotus	(Spreadsheet)
Memex Information	(E-pub)
Metcet Research	(CIM)
MicroFocus	(Dev Environment)
Micronim	(Database)
Mihalism Associates	(Visualisation)
Natural Language	(CASE)
Natural Language	(Database)
Netlabs Inc.	(Networking/Comm)
Numerical Design	(Visulisation)
Oracle Corp	(Database)
P-Stat	(Data Analysis)
Quintus	(AI)
Reasoning Systems	(CASE)
Research Systems Inc.	(Visulisation)
Saber Software	(CASE)
SAS	(Data Analysis
SCO Inc.	(Visualisation)
Sherrill Lubinski	(Visualisation)
SoftQuad	(E-pubs)
Softscience	(CASE)
Sybase	(Database)
System Strategies Ltd	(Interface)
UNICAD	(CASE)
Unify Corp.	(Database)
University of Pennsylvania	(Medical)
V-Systems Inc.	(Fax Software)
Wolfram Research	(Data Analysis)
WordPerfect	(Text Processing)

Figure 5: OPEN LOOK GUI ISV Commitment

Name Change

UNIX System Laboratories Europe Ltd, formerly AT&T UNIX Software Operation Europe, has responsibility for the sale and marketing of UNIX System V and associated products throughout Europe. The company is a subsiduary of UNIX System Laboratories Inc, a wholly owned subsiduary of AT&T.

All the UNIX trademarks that were formerly owned by AT&T have been transferred to USL Inc. These include UNIX, TUXEDO, OPEN LOOK, XWIN, Writers Workbench, Instructional Workbench and Documenters Workbench.

They are registered trademarks of the UNIX System Laboratories Inc in the USA and other countries.

C++ Column

Mark Rafter
rafter@warwick.uucp

Warwick University

Mark Rafter is a lecturer in Computer Science at Warwick University. He has been using C++ in research and teaching since Jan 1985. At present his research interests involve the use of parameterised classes and multiple inheritance to build a class library for use in hyperbolic geometry.

The Annotated C++ Reference Manual,
Margaret A. Ellis and Bjarne Stroustrup,
Addison-Wesley,
ISBN 0-201-51459-1,
(UK) Hard cover Price £31.45,
UK publication date: late August 1990

The book starts:

> ''This book provides a complete language reference manual for the expert C++ user.''

This is accurate!

It is a comprehensive statement of what C++ is, why it is the way it is, why it isn't the way it isn't, how its parts relate to each other, how the language relates to its cousins, how aspects of the language can be implemented, and where the language is heading.

The book is not introductory or tutorial in nature – it contains no complete programs and no explanation of how the various language constructs should be used in producing well structured software. It is not aimed at the novice or the casual C++ programmer. It is a reference manual for language specialists, C++ experts and for serious users of C++ that wish (or need) to obtain a complete grasp of the language.

The book contains the current definition of the C++ language and is the document that ANSI have taken as the starting point for the C++ standardisation work. It has 18 chapters in 408 pages, a 39 page index and is extensively cross-referenced. It is organised as an expanded form of the C++ Language Reference Manual that was distributed with release 2.1 of C++ from AT&T. (This, in turn, is a corrected form of the Draft C++ Reference Manual that was distributed with release 2.0 – which in its turn started life as the Reference Manual chapter (pages 245-311) of Stroustrup's previous book "The C++ Programming Language").

Each section of the AT&T 2.1 C++ Reference Manual expands to become a chapter in the book. These chapters combine three types of material: the reference manual proper, commentary interwoven with this reference material, and major discussions which are placed at the ends of chapters. The typesetting clearly distinguishes commentary from the reference material upon which it is commenting.

The reference material proper is almost exactly the same as the material in the AT&T 2.1 C++ Reference Manual (the differences are discussed below). Taken by itself this material is dry and leaves the interested reader asking many questions of the form "Why does the language allow this rather than that?".

The interwoven commentary answers many of these questions where they naturally occur, sometimes including examples of the consequences of alternative design choices. This will often cause readers to react: "Oh! I hadn't thought of that."

The commentary often presents examples of languages constructs in the form of program fragments together with explanatory text. Among many other things, the commentary is used to emphasise important points, to highlight fine distinctions, to draw the reader's attention to implications that may be easy to overlook, to clarify points in the reference material that may be thought to be ambiguous or vague, to give examples of things that are not in the language, to discuss the relationship to ANSI C, Classic C and earlier variants of C++, to discuss implementation issues, and occasionally to orient the the less-experienced reader.

Major discussions are placed at the end of thirteen of the chapters. These include discussions of the history of the language, its design philosophy, portability considerations and future languages features. Also covered are implementation techniques for overloaded functions, pointers-to-members and multiple inheritance. Much of this material has been published elsewhere, some of it is of independent interest – almost all of it is necessary for a deep understanding of the language and its spirit.

Templates and Exception Handling are the future language features. They are neither part of the language definition nor of any widely available implementation. Two chapters composed entirely of commentary describe the current proposals in these areas.

Earlier this year I had gone through the 2.0 Draft Reference Manual with a fine tooth comb, so I was familiar with a previous version of the unannotated material. That had been done in a "random access" fashion and had been hard work.

When I picked the up the book I was a bit daunted by its size, but in fact about 50% of it is explanatory commentary. The interweaving of commentary and reference material seems to have kept my attention far better that the raw reference material had done previously. I found that I could quite happily read from the front to back without a lot of page thrashing for forward and backward references. I attribute this in part to the careful location of the commentary which is full of interesting items and relevant examples.

The language itself has been cleaned up in places. For example, initialisers for references must not introduce temporaries so a reference-to-int can no longer be initialised with the constant 1 you must use a reference-to-const-int here instead. This is an important change and it naturally provokes the question: "How does my C++ system relate to the language defined by the book?". Of course this is not a question that the book should answer – each implementation must answer it, preferably by relating it to the language definition reference material in the book.

For AT&T C++ 2.0, the major differences from the book are:

- the introduction of true nested classes,
- deleted arrays no longer specify their length,
- typedefs and enumerations nest within classes,
- initialisers for references must not introduce temporaries,
- a pointer to constant may not be deleted,
- protected base classes are allowed,
- default constructors may have default arguments,
- the increment and decrement operators may have distinct prefix and postfix definitions.

At present there is no implementation of C++ that fully implements the language defined by the book – AT&T C++ 2.1 comes close but does not implement the last three changes. However, the Release Notes that accompany AT&T C++ 2.1 are recommended reading. These give an extremely clear statement of the relationships between AT&T's 1.2, 2.0 & 2.1 languages and implementations, and how migration from 2.0 to 2.1 will be managed.

Unfortunately, the AT&T C++ 2.1 Language Reference Manual is *not quite* identical to the reference material contained in the book – for example see 8.2.2 (arrays of references). In all cases where I checked it was the book that was correct.

I finished the book with a better understanding of the language than I had started with. This is due in part to the cleanup the language has undergone, but it is mainly due to the strengths of the book itself. It is well organised, contains very few typos, and is well produced. The major commentaries gather together important material that has been scattered through the literature and the interwoven commentary transforms the important, but dry, reference material into a hard, but interesting, read. Almost all the target readership will learn something from this book – most will learn a lot.

STOP PRESS

News has just reached me that the ANSI C++ standardisation committee has voted to accept the changes to C++ enshrined in the "Annotated C++ Reference Manual" into the draft standard. In addition the chapter on templates has also been accepted.

!%@:: A Directory to Electronic Mail Addressing and Networks Second Edition, 1990

The new 1990 edition of !%A:: A Directory of Electronic Mail Addressing and Networks, by Donnalyn Frey and Rick Adams is now available. This new edition provides readers with a directory and usage guide to approximately 130 of the world's research and educational networks, as well as commercial networks. The network information has been updated for 1990, with many new networks added. The directory makes it easy for readers to find networks they can use to reach other people around the world and guides readers in how to use them. It also assists readers in finding someone's email address and sending mail. The book is in an easy-to-use short reference format.

The directory is of use to system administrators who field electronic mail questions, network administrators who work with networks in other countries, researchers who want to get in touch with other researchers, conference attendees with many contacts, and others who routinely send email. Each network section contains general information about the network, as well as address structure and format, connections to other sites or networks, facilities available to users, contact name and address, cross references to other networks, network architecture, future plans, date of the last update, and a map showing the network location. Also included is a three-way index to network name, network type, and country, as well as a list of many of the world's second and third level domains.

This new edition contains:

- information on new networks such as AlterNet, CANET, CA*net, EASInet, InterEUnet, IXI, MFENET-II, TUVAKA, XLINK, and YNET
- updated information on networks that are reorganising or have reorganised, such as BIONET, ESNET, MFENET, NYSERnet, and OnTyme
- information on networks in the Soviet Union, Eastern European countries, and the People's Republic of China
- networks not in the first edition, such as ATT Mail, KREOnet, and SCIENCEnet
- updates of most of the existing networks described in the first edition which was published in 1989.

Many networks have had significant service or architecture changes, as well as new network connections or connections to new computers since publication of the first edition. For example, AARNet, the new major Australian network, was in the planning stages when the first edition of the book was published. AARNet is now a functioning network with connections both within Australia and to other countries. As another example, the Canadian national TCP/IP network, CA*net did not exist in 1989. CA*net is beginning operations in 1990 and soon will be the major network in Canada.

Small, but significant changes occurred in many other networks, changes such as new higher-speed lines, new connections in the country, and more. For example, mcvax, the major central networking computer for EUnet in Europe, was renamed mcsun. In many cases, network administrators had changed, so the authors provide new contacts. The authors include the new connections to the Soviet Union, Eastern Europe, and the People's Republic of China. Many of the connections existed before for private use. Now, the connections are open and people are encouraged to contact colleagues in these countries.

This new edition is the most up-to-date guide for directing your electronic mail; it is a real time saver. The book will continue to be updated every ten to twelve months. Readers who fill out the response card in the book have the option of either receiving notification of updates or receiving the updated edition automatically at a 25% discount.

MAROSI

Marosi, founded in 1989 by Dr. Pamela Gray to help organizations improve their use of technology through Open Systems, announces three important products to help programmers and their managers keep abreast of developing Open Systems standards:

The Marosi Standards Review, a quarterly journal edited by Dominic Dunlop of The Standard Answer Ltd., a long-time participant in standardization activities. The Review will keep you informed of developments and trends throughout the field of Open Systems standards.

The C Portability Verifier, a product of Mindcraft, the company which wrote IBM's POSIX Conformance Test Suite, checks your C programs for conformance to ANSI C, ISO POSIX, and the interfaces specified in the X/Open Portability Guide, edition 3. Available in a variety of formats, prices start at £495.

How to write portable UNIX™ programs using POSIX, a two-day seminar authored by Mindcraft, will be run for the first time in Europe on 5th-6th September, 1990 at a location close to Heathrow airport, England. Attendees, paying £595 for a place, qualify for a £50 discount on the C Portability Verifier.

Mindcraft and C Portability Verifier are trademarks of Mindcraft Inc.; X/Open is a trademark of the X/Open Group Members; IBM is a registered trademark of International Business Machines Corporation; UNIX is a registerd trademark of UNIX System Laloratories in the U.S.A. and other countries

✂ -

☐ Please send me a free sample copy of the Marosi Standards Review

☐ Please send further details of the Mindcraft C Portability Verifier

☐ Please send further details of the Mindcraft seminar *How to Write Portable UNIX Programs Using POSIX*

☐ Please enroll me on the first European *How to Write Portable UNIX Programs Using POSIX* seminar. I enclose a cheque or money order for £595

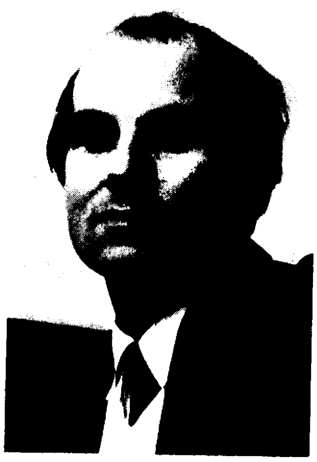
Marosi Limited
Scammell House
High Street
Ascot SL5 7JF
England

Telephone: +44 990 873155
Facsimile: +44 990 27328
Email: marosi@tsa.co.uk

Name: ____________________
Company: ____________________
Address: ____________________
____
____
Country: __________
Phone: __________
Fax: __________
Email: __________

MAROSI

Report on ISO/IEC JTC1/SC22/WG15 (POSIX)



Dominic Dunlop
domo@tsa.co.uk

The Standard Answer Ltd

Dominic Dunlop, an inveterate busy-body who likes to see order in all things – except on his desk-top – has been making a nuisiance of himself in UNIX standardization forums since the early nineteen-eighties, when he got tired of the needless differences between the many systems that were targets for his software porting activities. He would have gone to the recent UKUUG conference in London, and to the POSIX meeting in Danvers, but was busy getting married at the time...

Report on ISO/IEC JTC1/SC22/WG15 (POSIX) Meeting of 11th – 15th June, 1990, Paris, France

Introduction

Working Group 15 of Subcommittee 22 of Joint Technical Committee 1 of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC22/WG15), or, more briefly, the ISO POSIX working group, met in Paris, France, from the 12th to the 15th of June. The meeting was hosted by AFNOR, (Association française de normalisation), the French national standards body, at its offices in La Défense, a high-rise business district a brisk train-ride away from the city centre, and was preceded on 11th June and the morning of the 12th by meetings of the rapporteur groups on conformance testing, internationalization and security. Attendance was good, with thirty "experts" (as the *ISO Directives* style us) representing nine countries, plus the European Community.

I was present at the main meeting and at the internationalization rapporteur group as an observer with the brief of reporting back to you. This report is the fourth jointly commissioned by the European UNIX systems User Group (EUUG) and USENIX. As usual, it's a little imprecise in its references to ISO. Strictly, most of these should be to JTC1, or to some part of JTC1. Where precision is needed, I use it and give an explanation, but for the most part I refer simply to ISO, so as to avoid getting bogged down in unnecessary detail. If you have any comments, or need clarification or further information, please contact me at the mail address above.

First, a summary of the most important aspects of the meeting:

Summary

- POSIX.1, the operating system interface standard, should be published as international standard 9945-1 Real Soon Now. As I write, the ballot on the document has yet to close, but it seems unlikely that any of the twenty countries voting will oppose acceptance. The meeting identified a number of trivial editorial changes to the current draft international standard, and these, taken together with continuing nit-picking comments from ISO's central secretariat, should result in a document which ISO will print. Its technical content will be very close to that of

ANSI/IEEE Std. 1003.1:1988, so implementors following the U.S. standard can be assured of ISO compliance when 9945-1 finally sees the light of day.

- POSIX.2, the shell and tools standard, faces a bumpier ride before becoming international standard 9945-2. In an ISO ballot on acceptance of draft 9 of IEEE 1003.2 as a draft international standard, six countries voted against, with just five in favour. This is more of an embarrassment than a set-back: hindsight suggests that it was just too early to hold a ballot.
- Showing its increasing concern and frustration at the lack of apparent progress within the IEEE on a (programming) language-independent version of the POSIX operating system interface standard, WG15 has refused to "register" the current, largely language-independent, draft of the 1003.4 realtime extensions standard on the grounds that it makes little sense to have language-independent extensions to a base standard which is specified in terms of the C language. Similarly, the group failed to register 1003.5 (Ada binding) and 1003.9 (FORTRAN-77 binding) because POSIX.1 lacks an explicit service definition to which they can bind.
- The failure to register these documents causes a hiccup — albeit a discreet one — in the synchronization between IEEE and ISO POSIX standardization schedules. In the hope of avoiding such situations in the future, an informal "speak now, or forever hold your peace" mechanism has been put in place, allowing the international community to comment on the subject area, format, presentation and approach of IEEE documents at an early stage in their preparation.
- Had it not been for this upset, the working group would have adopted a firm synchronization plan so as to ensure that the work of IEEE and of ISO is closely coordinated in the future. As it is, the "U.S. member body" — ANSI — has been asked to provide a revised plan for the working group's October meeting in Seattle.
- The Open Software Foundation, UNIX International and X/Open are cooperating on a common approach to conformance testing, known as Phœnix. Governmental organizations with a strong interest in the field, such as the National Institute for Science and Technology (NIST) and the Commission of European Communities (CEC), seem broadly to welcome this move — even if the unaccustomed show of tripartite unity is, as one security rapporteur put it, "a bit spooky".
- At an evening reception hosted by AFUU (Association française des utilizateurs d'UNIX), the French UNIX users' group, Mike Lambert of X/Open said that "our hand is extended" to the international standardization community, with which his organization hopes to work in finding some more timely and responsive way of delivering formal consensus standards for open systems. The definition of this mechanism is left as an exercise for the reader — or for the international standards community. Perhaps Mike has come to realize that standardisers too are prone to the Not Invented Here syndrome, and so must believe that they thought of something themselves before they can accept it.
- Just to keep us all on our toes, ISO has updated its Directives, with the result that the procedure for turning a base document — such as one of the IEEE drafts — into an international standard is subtly changed. We now have to forget about Draft Proposals, and turn our minds instead to Working Drafts and Committee Drafts. Sigh...

The rest of this report gives more detail most of these topics.

9945-1 — Operating System Interface

You may recall from my report of WG15's last meeting in October, 1989, that the group had in effect told ISO's central secretariat that, while the then-current draft of IS 9945-1 was not perfect, it was, in the group's opinion, good enough to publish, particularly since we'd undertake to fix things up on short order, allowing an improved version to be published within a year of the first edition.

This proposal did not fly. Firstly, the central secretariat (now dynamically known as ITTF, the Information Technology Task Force), refused to publish a document that looked much more like an IEEE standard than an ISO standard; secondly, they deemed the changes needed to polish up the draft to be sufficiently great that it should go out to ballot again in order to provide a formal check that it was still acceptable to the group. This was duly done; the ballot closed on 29th June, and is expected to pass very comfortably.

Nevertheless, the ballot gave people the opportunity to comment formally on the document again, with the result that a number of small bug-fixes and clarifications were suggested, and were accepted for incorporation into the draft. We judge — and we hope that ITTF agrees — the changes are strictly editorial, and so will not merit yet another ballot. ITTF, which, in discussion with the IEEE, had slightly bent its drafting and presentation rules so as to come up with a compromise satisfactory to both parties, also suggested further changes, some in areas that we thought had already been addressed. This is a cause for concern: the prospect of the standard never being published because its layout and language do not meet some ill-defined guidelines does not appeal. Consequently, we are seeking "written harmonised editorial requirements" from the IEEE and ITTF.

The ballot also produced a number of suggestions in the area of internationalization, such as how to handle (and indeed, how to refer to) wide, or multi-byte, characters. To have acted on these comments would have changed the technical content of the draft standard — the equivalent of sliding down a snake in the game that leads to eventual publication. Happily, those who suggested the changes were content to leave these issues for the second edition of the standard, rather than further delay the first edition.

The single exception that we could get away with was to replace Annex E's[1] example international profile for the hypothetical (and extremely odd) land of Poz with a real example for the (only slightly odd) country of Denmark. Although this is a large change, it can be made because the annex (ISO-speak for appendix) is "non-normative"; that is, it is an explanatory example rather than a part of the official standard.

Messaging, an issue which is currently exercising the minds of those concerned with the next edition of the 1003.1 standard[1], was also passed over by WG15: nobody had a strong preference for either the X/Open proposal or the UniForum proposal, so 1003.1 will have to handle the issue without guidance from from the ISO working group.

1. This material is not in the published IEEE std. 1003.1:1988.

Where all does this leave us? With no published international standard as yet, but with a very good prospect of having one this year. Until it arrives, keep using the bilious green book, IEEE std. 1003.1:1988[2], as its technical content is very close to that of the eventual ISO standard[3].

9945-2 — Shell and Tools

Earlier in the year, there had been a ballot on moving forward draft proposal (DP) 9945-2, *Shell and utility application interface for computer operating system environments*, to become a draft international standard (DIS). Basically, while a DP is allowed — even expected — to differ considerably from the international standard which ultimately results, a DIS is expected to be in a format and to have contents which are very close to those of the ultimate standard[4]. That the ballot was six against to just five for (with nine countries not voting) indicates that the consensus is that 9945-2 has to go through quite a few more changes before it can be acceptable as an international standard.

Many of these changes concern internationalization, as this topic affects POSIX.2 considerably more than POSIX.1. For example, the example Danish national profile to be incorporated into 9945-1 is just a part of the work that DS (Dansk Standardiseringråd) has done on the topic; the rest affects 9945-2. There is also an unresolved issue concerning the definition of collation sequences (sorting orders) for

2. You can buy a copy by calling IEEE customer service on +1 908 981 1393 (1 800 678 IEEE inside the U.S.) and giving them a credit card number. The cost is $37, plus $4 for overseas surface mail, plus another $15 for airmail. Alternatively, your national standards body may be able to sell you a copy. For example, BSI in the U.K. has them for sale at £24.

3. A new edition of ANSI/IEEE 1003.1, to be published this year, will be identical in technical content to the ISO standard.

4. DP 9945-2 is the last that we will see; under the new directives, DPs are no more; they are replaced by working drafts (WDs), which become committee drafts (CDs) before becoming DISs. This is not a big deal.

international character sets. Utilities such as *sort* clearly need to know about collation sequence, and so do the regular expression-handling utilities and functions defined by POSIX.2. The problem is that nobody in ISO seems to want to handle the matter. In particular, JTC1/SC2, which standardises coded character sets, sees its job as assigning codes to characters, not as saying anything about how those codes should sort[5]. This is a reasonable attitude: collation is a surprisingly complex field, and to attempt to cover it in coded character set standards would break the ISO rule of one topic, one standard. This is all very well, but 9945-2 needs somebody to do the work, and WG15 may end up doing it itself if pleas for help from elsewhere in ISO fail.

More work is on the way: 1003.2a, the User Portability Extension to POSIX.2, was registered for ballot as a Proposed Draft Amendment (PDAM) to DP 9945-2, giving the international community a chance to say what it thinks of the idea of standardizing interactive commands such as *vi* and language processors like *cc* — or rather *c89*.

Coordination

The coordination arrangements which will make IEEE and ISO work on POSIX march forward in lock-step were almost complete before the small international rebellion on the matter of language independence made us stumble. (See below.) Because WG15 failed to register 1003.4, 1003.5 and 1003.9 at this meeting, the plan must be adjusted, although little slippage should result. We'll try to jump on board the revised plan at the next meeting.

Internationalization

In the one and a half days before the main meeting of WG15, its three rapporteur groups met. I attended the internationalization meeting, which had a hectic time of it: as the only rapporteur group directly concerned with the current content of 9945-1 and -2, we had a lot of comments to sift through prior to the main meeting. This we did, by the skin of our teeth. Our conclusions are largely reflected in the actions on the two drafts, reported above.

Security

The security rapporteur group is just getting off the ground. As with internationalization, activities scattered throughout JTC1 have to do with security. But in contrast to the current situation for internationalization, for security there is a (very new) subcommittee, SC27. Conceivably, SC27 could define some all-encompassing ISO security model[6] which would not suit POSIX and the work of 1003.6, which is eventually to be integrated into the 9945 standards. The rapporteur group is doing all that it can to prevent this from happening. Happily, ISO is already aware of the issue, and has formed a special working group on security, to which WG15 will be sending a representative.

Conformance Testing

The proceedings of the rapporteur group on conformance testing were rather overshadowed by the announcement of Project Phœnix. Quite from what ashes this has risen we did not learn; however, as it involves cooperation between the Open Software Foundation (OSF), UNIX International and X/Open, it is difficult to resist the temptation to speculate. But resist I shall...

The goal of Phœnix is to develop a common conformance testing suite and methodology for the three organizations, or, to put it another way, to harmonize their activities in this area. Harmonization of standards for open systems is an important issue for WG15 in general. The issue affects the rapporteur group on conformance testing in particular because the European Community and NIST are giving a high priority to accrediting test houses which can check conformance to open systems standards. This means that they need to ensure that uniform test methods are being consistently applied. The rapporteur group will address this issue at its next meeting. In the mean time, WG15 has registered the current draft of the first part of 1003.3, which describes general test procedures, and we will

5. For oblique confirmation from "the father of ASCII", see [2].

6. ISO likes models, and they're not without their uses. Work on a useful model for open systems is under way in several forums.

review it before our next meeting — despite the fact that there is as yet no well-defined route by which POSIX.3 can become an international standard.

Language Independence

The issue of a making the POSIX standards independent of any particular computer language came to the fore at this meeting. What's all the fuss about? Well, ISO has firm — and, in my view, sensible — ideas about how to write standards which define the services available from information processing systems. Building on the doctrine that one standard should address just one topic, there should be, says ISO, one document defining the service, and one or more documents describing ways of accessing that service. For example, a browse through the open systems interconnection standards reveals several groupings such as IS 8072, *Transport Service Definition* and IS 8073, *Connection oriented transport protocol specification*; and IS 8649, *Service definition for the Association Control Service Element*, and IS 8650, *Protocol specification for the Association Control Service Element*[7]. Similarly, in text communication, there is IS 9066-1, *Reliable transfer — model and service definition* and IS 9066-2, *Reliable transfer — protocol specification*, and in graphics, IS 7942, *Graphical Kernel System functional description* and IS 8651-1 through -3 giving GKS language bindings for FORTRAN, Pascal and Ada. (8651-4, giving bindings for C, is in the works.)

POSIX, ISO has ordained, must eventually go along with this model, with the result that IS 9945-1, -2, and -3 (Operating system interface, shell and utilities, and administration respectively) will be concerned simply with defining services, and will not be bound to any particular programming language. The key word here is "eventually": because of the pressing need for an international standard for POSIX, a waiver has been granted, allowing the first version of the 9945-1 and -2 standards to be a combination of (purists might say "a collision between") a service definition and a C language binding to those services. The expectation is that a future revised new edition of each standard will be language-independent, and that bindings for the C language will be published as a separate standard at the same time[8].

So far, so good. But this is where the problems start. While this mandated future appears a long way off if one looks at the IEEE's work on POSIX.1, it seems very close when POSIX.4 (realtime extensions), POSIX.5 (Ada bindings) and POSIX.9 (FORTRAN-77 bindings) are considered. In the case of POSIX.4, language-independent extensions are being proposed for a standard which is not itself (yet) language-independent. And POSIX.5 and POSIX.9 define bindings to a set of services which is not explicitly defined, but rather is defined implicitly in terms of the binding to the C language given in POSIX.1. In the absence of a clear service definition, it is no surprise that, for good reasons which have to do with their respective languages, the Ada, C and FORTRAN versions of the operating system interface appear currently to be binding to slightly different sets of services.

To some, the whole issue of language independence seems like an unnecessary and irksome imposition by ISO. In a recent posting to comp.std.unix[3], Doug Gwyn wrote:

> [Those of us who worked on 1003.1] did NOT set out to create a language-independent standard; C was specifically chosen for the obvious reason that it was the SOLE appropriate language for systems-level programming on UNIX, for a variety of reasons, including the fact that the UNIX kernel has a marked preference for being fed C data types.

It is certainly true that, because, back in 1985, all the base documents for the IEEE POSIX work were written in terms of a C language interface, the working group made a good pragmatic decision to

7. Browsing through OSI standards may be a cure for insomnia. On the other hand, it may aggravate hypertension...

8. Under ISO's procedures, the bindings standards for POSIX will be allocated an ISO standard number just as soon as we register a base document for one of them. Until that happens, I shall have to refer to "future bindings standards", rather than having a convenient number to use as a handle.

produce a standard centered on C. A different decision would have resulted in the standard which took longer to get out of the door, and which would not have been any more useful to its primary audience — committed UNIX users concerned with the divergence among implementations of their chosen operating system. But the success of UNIX and of its offspring, POSIX, has greatly widened the audience for the standard. Whether we like it or not, ISO has revisited the original decision so as to ensure that the international standards for POSIX meet the needs of this new audience. As a result (to continue quoting from [3]):

> This "language binding" nonsense was foisted off on P1003 in an attempt to meet ISO guidelines. I think it must have been adopted by ISO as the result of Pascal types insisting that they never have to use any other language.

Countering this, I would contend that, while the number of "Pascal types" is too small for their opinion to be of prime concern, the number of FORTRAN types, COBOL types and perhaps even of Ada types is large enough that it would be at least polite to provide some well-defined means whereby these communities can create bindings which allow them to hook into POSIX services without having to learn a new programming language. In the future, the growing C++ community may decide to define the interface to POSIX services in an object-oriented manner; Steve Carter paid us a flying visit with news from the ANSI X3J16 C++ committee in order to open up channels of communication.

Consider another topic which has come to the fore as POSIX has moved into the international arena: internationalization — mechanisms which will allow non-English speakers to use POSIX-compliant systems without having to learn a new natural language. Like the current movement towards separating service definitions from bindings, this involves a considerable amount of work, yet does not appear to provide much that is of use to UNIX' original community of technical users. Accommodating the preferences (prejudices?) of ever greater numbers of people is, it seems to me, part of the price of success for the UNIX operating system. And it may well pay dividends. For example, internationalization work on regular expressions and collating has resulted in facilities which will be of use even to English speakers.

Returning to the matter of the programming language used for bindings, it is true that AT&T-derived UNIX implementations prefer a diet of C data types. However, it certainly was an aim of 1003.1 to allow hosted POSIX implementations, which might well be riding on underlying operating systems with entirely different tastes. As a topical example, lightweight kernels such as Chorus and Mach live on messages, suggesting that their services could be bound to a data stream encoding[9]. I suspect that anybody who has tried to make *ioctl*( ) work across a network wishes that UNIX had anticipated their needs by following such a model from the start. But it didn't, and to redefine it in these terms would be a large piece of work which (thankfully) is a long way beyond the scope of WG15.

There is no way in which all such requirements could have been anticipated, and accommodating the most important of them as the need arises inevitably causes pain. Both language independence and internationalization are unanticipated requirements which the international community wants retrofitted to POSIX on short order. And it's ANSI, as provider of the base documents to ISO, and the IEEE, as the body accredited by ANSI to produce the documents, that get beat on to do the real work, and to suffer the pain.

In the view of WG15, the real work needed to make POSIX.1 a logical base for extensions such as POSIX.4, POSIX.5 and POSIX.9 is not being done fast enough. Trouble is, all standards are produced by volunteers — often volunteers who have had to make a case to their managements that there's some percentage in their company being involved in standards work. There is clearly an eventual percentage in language independence for suppliers of POSIX-conformant systems if it encourages users of languages not traditionally found on UNIX systems to migrate to POSIX. But sadly, while not in any way

9. More ISO-speak: broadly, if you have a protocol that lives above layer five (session) of the OSI stack, you'd better call it a data stream encoding. For example, the protocol for the X Window System™ is a data stream encoding by this definition.

criticizing the quality of the work done to date, there aren't enough IEEE volunteers interested in recasting POSIX.1 into language-independent form.

Maybe, just maybe, if the international community is more interested than the U.S. in getting this work done, WG15 should encourage more people from outside the U.S. to participate actively and directly in the work of the IEEE. (Or, to put it another way, encourage more organizations outside the U.S. to put their hands more deeply into their pockets in order to pay for people to attend IEEE POSIX working group meetings.) The alternative is that WG15 does the work itself — an alternative I'd rather not contemplate.

For now, two action items on ANSI from WG15 sum up the situation:

> Pursue with vigour the production of a language-independent version of both 9945-1 and P1003.4 in conjunction with a C language binding for each in order that they are eligible as replacements for 9945-1:1990.
>
> Request the IEEE to expedite the completion of the language independent specification of 9945-1 that is precisely functionally equivalent to the existing 9945-1:1990 and provide a C language binding that is syntactically and semantically identical; and request that a detailed proposal status report on this issue including a synchronization proposal be presented at the next meeting of WG15.

Next Meeting

The next meeting of WG15 is in Seattle from 23rd to 26th October — the week after the IEEE POSIX working group meeting in the same city (and the same week as the EUUG meeting in Nice, France[10]). Should be interesting!

References

1. June, 1990 Standards Update, Jeffrey S. Haemer, comp.std.unix Volume 20, Number 66, USENIX, 30 June 1990

2. Letter from R. W. Bremer, pp 34-35, *Byte*, volume 15, number 6, June 1990

3. Doug Gwyn, comp.std.unix Volume 20, Number 51, USENET, 27 June 1990

10. In two meetings, WG15 has managed to clash both with summer USENIX and with autumn EUUG. It almost looks as if we do it on purpose! But we don't, and will try to do better next year...

CONFERENCE ANNOUNCEMENT

UNIX in Deutschland
Wiesbaden
3-6 September 1990

This conference is held together with the annual meeting of the GUUG, together with the largest UNIX Exhibition in Germany. The exhibition has grown from last years event by 38% and uses 2,400 square meters (net.) More than 900 attendees are expected for the conference. There will be one day with tutorials followed by the conference with three sessions (two technical, one commercial) in parallel. The conference language is primarily German. Here, briefly, is the conference outline:

Monday, 3rd	Tutorials	
	T1	OSF/1
	T2	Intr. to Programming with Motif
	T3	EUnet
	T4	UNIX System V, Rel 4.0 for Develp.
	T5	UNIX Software Engeneering
Tuesday, 4th	Introduction	
	Invited Speakers	
	Session 1	Operating Systems
	Session 2	Applications
	Session F1	Software — Tools and Methods
	Session F2	System Management
	Panel session	Manufactures and Open System
	GUUG Annual Meeting	
Wednesday, 5th	Invited Speakers	
	Session 3	Object Oriented Languages
	Session 4	Tools
	Session 5	Data Management Systems
	Session 6	Communication
	Session F3	Retrieval Systems
	Session F4	Graphical User Interfaces
	Social Event	
Thursday, 6th	Invited Speakers	
	Session 7	Security
	Session 8	Secure UNIX
	Session 9	UNIX and CIM
	Session 10	User Interfaces
	Session F5	Distributed Systems
	Session F6	Hard- and Software Architectures

Detailed Information can be requested via the GUUG office in Munich:

Elsenheimer Str. 43
D 8000 Munich 40
W Germany
+49-89-5707697
guug@unido.uucp

GUUG/East

GUUG/east has been founded April 23th in Leipzig (GDR). There is a strong relationship and cooperation between GUUG and GUUG/east. It is planned to merge the two groups into one after the merge of the two German states. Details on this will follow later.

Puzzle Corner

Mick Farmer
mick@cs.bbk.ac.uk

Birkbeck College

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

Hello peeps,

Solution to Puzzle Number 12

The force pressing the tile against the roof is 2 × cos 20° and, as the coefficient of friction is 0.5, the total frictional force at incipient motion is 0.5 × 2 × cos 20°, or 0.9397 lbs. The force causing sliding is 2 × sin 20°, or 0.6840 lbs.

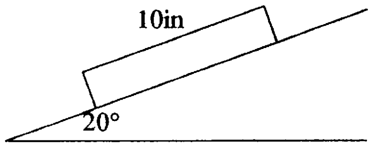


Figure 1 – The Creeping Roof Tile

When thermal expansion causes motion with respect to the roof, the total frictional resistance of 0.9397 lbs. must be developed, yet in general the tile will not move so that the overall force is 0.6840 lbs. These conditions are met if the difference (0.2557 lbs.) occurs half in one direction and half in the other. Thus a frictional resistance tending to shove the tile down the roof of 0.12785 lbs. is developed on the heating cycle by the part of the tile above the centre of expansion. This is opposed by a frictional force of 0.81185 lbs. (0.6840 + 0.12785), developed on heating, by the part of the tile below the centre of expansion, tending to hold the tile in place. The centre of expansion is 1.361" (0.12785 / 0.9397 × 10) below the top of the tile. On expansion the centre of the tile moves down 0.00109" ((5 - 1.361) × 6 × 10^{-6} × 50). On contraction, the centre of contraction is 1.361" from the lower end of the tile and the centre creeps down an additional 0.00109" making the daily movement 0.00218".

Solution to Puzzle Number 13

To fully utilise the toaster, both sides must do an equal share of the work. Three sides of the bread slices must be toasted on each side of the toaster:

0.00 - 0.05	Put A slice in left
0.05 - 0.10	Put B slice in right
0.55 - 0.57	Turn slice A
0.60 - 0.65	Remove slice B
0.65 - 0.70	Put slice C in right
1.07 - 1.12	Remove slice A to plate (done)
1.12 - 1.17	Put slice B (second side) in left
1.20 - 1.22	Turn slice C
1.67 - 1.72	Remove slice B (done)
1.72 - 1.77	Remove slice C (done)

Challenging Quickies

For this issue, a slight change in format for the puzzles. You're invited to keep track of the time taken to reach the solution of these three *quickies*. This should not exceed five minutes each, since they are chiefly a test of mental agility in finding the best short-cut to the answer.

Puzzle Number 14

A 1" cube has its outline constructed out of wire whose resistance is one ohm per inch. What is the resistance between opposite corners of the cube?

Puzzle Number 15

Suppose I think of a number whch you are to determine by asking me not more than twenty questions, each of which can be answered by only "yes" or "no". What is the largest number I should be permitted to choose so that you may determine it in twenty questions?

Puzzle Number 16

Why must a house whose rooms each have an even number of doors also have an even number of outside entrance doors?

Loads-a-puzzles,

Mick

Call Doc Strange

Colston Sanger
doc.strange@gid.co.uk

GID Ltd

Three months have passed and Colston Sanger is still a senior consultant/tea boy with GID. He's also still a visiting lecturer in the Faculty of Engineering, Science and Mathematics at Middlesex Polytechnic. So what's new? Nothing really, except that he's been editing a book with Tony Elliman of Brunel: *Open Systems for Europe: towards 1992* (Chapman & Hall, forthcoming).

We Are Under Attack!

Err...yes, well. Following an incident at a gateway machine near me, it seemed like a good idea to discuss system security. Not at the usual level of 'Here are a thousand Berkeley bugs that a bored undergraduate cracker might be ingenious enough to use to break your system', but at the much more mundane level of the thousand and one silly things that system administrators themselves do that compromise the security of their systems. What I have to say is likely to be no more than plain common sense, and is aimed primarily at people who are relatively new to UNIX running a standalone System V machine with, at most, a dial-up connection to the outside world: people who administer large networks of Sun workstations are likely to find this stuff 'old hat', but not irrelevant.

Let me start at the beginning, with an extract from Captain's Log for Saturday, 16 June: [1]

```
********** qwerty **********
uucp qwerty  (6/16-15:36:44,10868,0) OK (startup)
uucp qwerty  (6/16-15:36:57,10868,0) REMOTE REQUESTED (sixnine!/etc/passwd
--> qwerty!/usr1/bloggs/rje/.sixnine.pw (fred))
fred qwerty  (6/16-15:36:57,10868,0) PERMISSION (DENIED)
uucp qwerty  (6/16-15:37:41,10868,0) REMOTE REQUESTED (sixnine!/usr/lib/
uucp/Systems --> qwerty!/usr1/bloggs/rje/.sixnine.sys (fred))
fred qwerty  (6/16-15:37:41,10868,0) PERMISSION (DENIED)
uucp qwerty  (6/16-15:37:56,10868,0) OK (conversation complete tty11 76)
```

I picked this up in the evening when mail messages generated by the standard System V UUCP admin daemons (specifically, `/usr/lib/uucp/uudemon.admin` and `/usr/lib/uucp/uudemon.cleanu` run from root's `crontab` file) showed up in my mailbox. Parenthetically, I also noticed that the gateway's modems were rather busy that evening.

Now, I know who 'fred' really is and, as it happens, I had asked him the day before to send me a couple of files from his machine, which is hidden behind the gateway. So my first response, in all innocence, was to send him a somewhat laconic mail message:

1. In what follows, all names have been changed to protect guilty parties.

```
From colston Sat Jun 16 23:24:10 1990
Subject: You don't *honestly* think...???
To: fred@qwerty
Date: Sat, 16 Jun 90 23:24:10 BST

Fred,

Why do you want sixnine's passwd and /usr/lib/uucp/Systems files?
Haven't you updated uipido since the great 071/081 changes?

Colston
```

Minutes later, on second thoughts, I cancelled it.

Next morning, Sunday, I called him at home. No, he hadn't requested any such thing ... At that point, I rang the gateway's system administrator and, to cut a long long story short, he spent most of a sunny Sunday afternoon in the office. As for fred, regardless of whether he had a password before, he certainly had one on Monday morning.

The incident is trivial in itself. The cracker wasn't able to get anything from my system, first because of normal UUCP permissions (in `/usr/lib/uucp/Permissions`) let alone file access permissions and, secondly, because I run AT&T System V Release 3.2 which implements a shadow password scheme. However, what happened to me also happened to all the other systems with which the gateway has links — with, I hasten to add, the same null result. That's not to say no result. Over the next several weeks, but mainly at weekends, the cracker tried repeatedly to gain access (all logged in `/usr/adm/loginlog`), disrupted dial-up communications and generally behaved like a sort of electronic lager lout — not so much your ordinary 'breaking and entering' style of cracking, more like malicious vandalism. Much more important, the incident caused a lot of irritation and embarrassment for the system administrator and the organisation concerned.

So, apart from the fact that at least one cracker out there has also been reading Clifford Stoll's *The Cuckoo's Egg*,[2] what are the lessons to be learned from the incident?

2. London (The Bodley Head), 1989, ISBN: 0 370 31433 6. Strongly recommended and a good read too.

Using And Choosing Passwords

First, it's not really a good idea to have user logins on a gateway machine. If you can, regard your gateway as a sort of 'fire-door' between the outside world and your operational systems. If you must have user logins for mail-stops or mail forwarding, then make sure they are blocked (with a '`*`' or some other uncryptable string in the password field in `/etc/passwd`) or have reasonable passwords.

What is a reasonable password? It's certainly not 'fred', 'wumpus', 'sh*thead' (*without* the asterisk), 'susan', 'jennifer' or any of the other common choices.[3] These are all Berkeley passwords, but the System V-mandated 'fred01' is hardly any better.

There are lots of *DON'T*'s in the literature for choosing passwords:

- *don't* use your login name or your first or last name in any form (reversed, capitalised, doubled, *etc*)
- *don't* use your significant other's or child's name

3. See Daniel V.Klein, 'Foiling the Cracker: A Survey of, and Improvements to, Password Security', *UKUUG Summer 1990 Conference Proceedings*, London, 9-13 July 1990, pp 147-54. Klein reports the results of a password cracking exercise on a sample database of 15,000 accounts: '21% (nearly 3000 passwords) were guessed in the first week, and ...in the first 15 minutes of testing 368 (or 2.7%) had been cracked...On an average system with 50 accounts in the `/etc/passwd` file, one could expect the first account to be cracked in under 2 minutes, with 5-15 accounts being cracked by the end of the first day.' He adds: 'Even though the `root` account may not be cracked, all it takes is one account being compromised for a cracker to establish a toehold in a system.'

- *don't* use easily obtainable information about yourself, such as your telephone, car registration or social security number or the name of the street where you live.

DO's, on the other hand, are a little harder to come by. Some good choices for passwords are:

- the first letter of each word from a line of a phrase, song or poem. For example, 'Ten green bottles standing on the wall' becomes '10gbsotw'; and 'You think I'm stupid?' becomes 'Ut1ams?'; 'On the first day of Christmas' becomes 'Ot1doC' — but you can make them up just as well as I can.
- two short words concatenated with one or more punctuation marks. For example, 'mad+beef' or 'Me?Mug?'.
- alternate between one or two consonants and vowels plus punctuation marks or numbers to make pronouncable, nonsense words. For example, 'ekapitog++' or 'zatoich!'.

Note that just because a password must have at least six characters, that doesn't mean it can *only* have six characters. Although only the first eight characters are significant, the longer a good password is, the harder it is to crack.

Whether you use password aging is up to you (in UNIX/386 it is on by default). Personally, I would encourage people to change their passwords from time to time, but not actually enforce it with password aging. Being faced with '`Your password has expired. Choose a new one.`' while still only half awake doesn't strike me as being conducive to choosing a good password.

One final point: in a commercial environment, the object of attack is as likely to be the financial director as it is `root`.

Physical Security

In System V, machines come up to the `initdefault` state, defined in `/etc/inittab` as either `2` (multi-user) or `3` (Remote File Sharing) so there is never that window of opportunity of single-user mode on the system console.

Then again, internal crackers can only login directly as `root`, should they discover the password, on the system console: on all other attached terminals they must login as themselves and then `su` to `root` — which means that there will be a record in the `/usr/adm/sulog` file. Even if they are unsuccessful, a record is still written to the file. If they try to `su` to another ordinary user, a record is written to the file. So you can investigate, can't you?

People who leave their terminals logged in while they go off to the pub at lunchtime or, even worse, leave their terminals logged in overnight or *over the weekend* are a menace: they deserve all they get by way of idle or time-out daemons.

As for the external cracker, take care with modem setup: misconfigured modems can fail to logout a user when the line drops — hence the next person dialling in, whoever they may be, can continue with the previous user's login session. (Be especially careful if you are logging in from home and `su`-ing to `root` to do remote system administration!)

Know Your System

I've talked in a previous column about a system administrator's basic toolkit:[4] half a dozen commands — `who -u`, `/etc/whodo`, `ps -ef` or `ps -efl`, `df` and `sar` — that if you get into the habit of running periodically throughout the day you can use to anticipate and head off problems before they become serious. Even a simple `ls -al` is useful for discovering all sorts of anomalies in the filesystem.

System accounting, sometimes used for charging people extortionate amounts of money, can equally well be employed for getting a feel for what is the 'normal' state of affairs on your machine, as well as for load balancing and detecting minor breaches of company policy such as playing `rogue` or `wanderer` during office hours. More on system accounting later.

One of the irritating things about System V is that there's no one place where log files are kept: you have to look in `/usr/adm` for some,

4. 'UNIX Clinic: No space on integral hard disk drive 0, partition 0', *EUUGN*, Vol.9 No.1 (Spring 1989); pp 63-68.

/usr/lib/cron for others, /usr/spool/uucp/.Admin for yet others and so on. I'm not sure if it's any better in System V Release 4.0 or in Berkeley UNIX, but at least there you've got syslogd. However, it's quite easy to make a simple-minded shell script to keep track of these things, such as the following:

```
# check_log
# Checks various log files for incursions/exceptions
ADMIN=colston
tail /usr/adm/loginlog | mailx -s "Loginlog Check" $ADMIN
tail /usr/adm/sulog | mailx -s "Sulog Check" $ADMIN

# Some of these will be reported by the UUCP daemons anyway
cat /usr/spool/UUCP/.Admin/Foreign | mailx -s "Who this?" $ADMIN
cat /usr/spool/UUCP/.Admin/errors | mailx -s "Trouble with UUCP" $ADMIN
cat /usr/spool/uucp/.Old/Old-Log-1 | mailx -s "UUCP Log" $ADMIN

# For smail - if you're that concerned
tail /usr/spool/uucp/mail.log | mailx -s "Mail.log Check" $ADMIN

# Who's printing what
tail /usr/spool/lp/logs/requests | mailx -s "Print jobs" $ADMIN
```

You'll need to run it out of the root crontab at a suitable time each night.

It is also a good idea to arrange for mail to root, adm, lp, uucp (and any of the other system logins that get status or error messages by mail) to be forwarded to yourself.

As for the really basic things, you should be aware of the *setuid* and *setgid* executables that are intended to be there. You can find them with:

```
find / -type f \( -perm -4000 -o -perm -2000 \) -print
```

Similarly, you can find files and directories that are writable by others (some shouldn't be) with:

```
find / -perm -2 -print
```

I guess that it would be as well to check the permissions on devices too – both in /dev and, if there are any, elsewhere.

And If You Are Attacked?

Don't fool about pretending to be Europe's answer to Dick Tracy. Enforce user password changes immediately and change your UUCP passwords. Very likely, your UUCP neighbours will also want to change the password that you use when calling them. If you are faced with persistent denial of service attacks — like the malicious vandalism I spoke of earlier — you may want to consider having the telephone numbers of your modems changed as well.

If the cracker has succeeded in breaking and entering, acctcom (part of the system accounting sub-system) invoked without any options can give you a complete action replay of all commands issued on your machine. With options, you can narrow down the search and the sheer bulk of output, as in:

```
acctcom -S 22:00 -u fred
```

where the -S option selects processes starting at or after 22:00 hours and the -u option is the user.[5]

5. Of course, a more sophisticated cracker than fred might not leave any traces...

Here is some sample output:

```
START AFT: Fri Jul 20 22:00:00 1990
COMMAND                        START    END             REAL      CPU    MEAN
NAME      USER    TTYNAME      TIME     TIME           (SECS)   (SECS)  SIZE(K)
who       fred    tty21        22:06:01 22:06:01        0.36     0.19     3.26
who       fred    tty21        22:06:04 22:06:04        0.25     0.16     3.88
df        fred    tty21        22:06:06 22:06:06        0.33     0.22     4.43
ps        fred    tty21        22:06:10 22:06:12        2.52     0.59     1.59
date      fred    tty21        22:06:16 22:06:16        0.22     0.11     5.82
su        fred    tty21        22:06:22 22:06:24        2.88     1.28     0.77
who       fred    tty21        22:08:10 22:08:10        0.33     0.20     4.20
sar       fred    tty21        22:08:15 22:08:17        2.82     0.72     1.69
ls        fred    tty21        22:08:21 22:08:21        0.67     0.37     4.32
cat       fred    tty21        22:08:34 22:08:36        2.95     0.26     2.46
sh        fred    tty21        22:08:50 22:08:50        0.16     0.11    22.00
#uucp     fred    tty21        22:08:50 22:08:51        1.39     0.62     2.29
uustat    fred    tty21        22:09:00 22:09:01        1.30     0.50     2.88
```

At the same time, if the damage is at all serious, you might as well brace yourself for a talk with 'the management'. (I know, nobody enjoys the sight of headless chickens running about, but they are going to find out sooner or later anyway.) Better find your distribution set and latest backup too, in case you have to do a complete reload.

What Do You Do If `root` Leaves?

What do you do if `root` leaves and you have to run the system yourself until you can find a successor?

First, don't start treating him or her as a potential criminal. After all, you are going to need help to understand how your system works. Before he or she leaves:

- go through `/etc/passwd` together so that you know who each login is
- similarly, go through `/usr/lib/uucp/Systems` so that you know and understand why you need dial-up links with each of the systems listed there
- have him or her explain how any customised, locally developed or third-party bits and pieces work from a system administration point of view.

After he or she leaves:

- change the `root` and other system passwords
- arrange to change UUCP passwords.

Security Is Your Responsibility

Let me end with a quotation from *Computer Weekly*, a UK trade paper, reporting on the first reading of the Computer Misuse Bill currently before the UK Parliament:

Computer owners with unreliable or insecure systems could face compensation claims for damage caused to individuals and an investigation by the Data Protection Registrar.

It goes on to quote the Labour Member of Parliament, Mr Harry Cohen:

"If we accept the premise implicit in the ... Computer Misuse Bill that computers have a special role in the running of our society and it is an offence to misuse a computer, we should also accept that a computer owner must have an obligation to take proper computer security precautions."

Bedtime Reading

The classic papers by Dennis Ritchie, 'On the Security of UNIX' and Robert Morris and Ken Thompson, 'Password Security: A Case History', often reprinted as part of the supplementary UNIX System documentation, are still well worth reading; as is F.T.Grampp and R.H.Morris, 'UNIX Operating System Security', *AT&T Bell Laboratories Technical Journal*, Vol.63 No.8 Part 2 (October 1984), pp 1649-1672. Most of the books on system administration have a chapter on security: Rebecca Thomas and Rik Farrow, *UNIX Administration Guide for System V*, Englewood Cliffs, NJ (Prentice Hall), 1989, ISBN 0-13-

942889-5 is as good as any and also has a useful chapter on system accounting; and Evi Nemeth, Garth Snyder and Scott Seebass, *UNIX System Administration Handbook*, Englewood Cliffs, NJ (Prentice Hall), 1989, ISBN 0-13-933441-6 covers Berkeley variants as well. *The* book on security is Patrick Wood and Stephen Kochan, *UNIX System Security*, Hasbrouck Heights, NJ (Hayden), 1985, ISBN 0-8104-6267-2 — only available in hardback and expensive, but worth it.

Unix for Users,
C D F Miller, R D Boyle, A J Stewart, Blackwell Scientific Publications, ISBN 0-632-02416 (UK) Price 12.95 UK Pounds, Soft Back, 250 pages including index,

Reviewed by Kelly Dunlop of Parliament Hill Computers Limited kelly@phcomp.co.uk

The blurb on the back of this book says it is intended "chiefly for readers who have not encountered Unix before" and as such it is a reasonably good book.

It starts off with a simple explanation of what is an operating system and what is Unix for users who are not familiar with computers. It then has a short section which basically says there are lots of systems out there that are called *IX and for the purpose of the book most of them can be classed as Unix. They explain they will try and point out where differences occur in the different varieties of Unix but this is not an easy thing to do and they do fall down in a few places. I like the comment "There are also several claimants to the title of Unix-like system which are, frankly, too far removed from Unix to deserve the name."

There is a section which describes "special characters" and this includes the delete character. It is a common mistake among new users to assume because the delete character is backspace on one machine it will be the same on every other one. It is nice to see a book explain that it is simply a setup feature and it may be different on each machine. The book then progresses to logging into Unix.

It then describes some simple commands to get started using Unix. These include cat, ls, rm, cp and mv. It also tells you how to help yourself on a Unix system (if someone has installed the on-line manual pages of course) by using the man command.

I would have thought pwd and cd belonged here but they are saved until the chapter describing files and directories. This is where they belong but maybe the order of chapters should have been such that the files and directories were explained first.

Now we move on to editing files and a chapter explaining the difference between ex and vi. It describes quite well how to go about creating a file using vi and how to edit it using example files. The command summaries are laid out in tables which makes them easy to find at a later date.

Next comes the chapter about Files, Directories and Users which describes directories, the tree structure of the Unix file system, protection modes and the super-user. This is well laid out and goes through things in an ordered manner.

There are a couple of fairly lengthy chapters describing the use of the Bourne Shell and C Shell as command and programming languages followed by a chapter on more advanced commands. The information contained in then could be found in the Unix manual set but is much easier to digest in this form

The rest of the book describes topics which I feel are more advanced and would be better tackled after the user has gained some experience with the commands he has already learnt.

Included here are C and the Unix Interface, Administration and Maintenance, Text Processing, Other Software, and Communincations and Networking. What concerns me is that a new user will pick up this book, race through it and assume he knows all there is to know about Unix.

Having said that it is one of the better books I have seen. The only criticism is that maybe it goes too far, too fast. It would be useful in an organisation where the "real" users have no time to train new users (common in many places), as a first step onto the Unix ladder.

Calendar of UNIX Events

This is a combined calendar of planned conferences, workshops, or standards meetings related to the UNIX operating system. Most of this information came from the various conference organizers, although some was taken from ;login: (USENIX), 13, 1, Jan/Feb 1988, CommUNIXations (/usr/group), VII, 6, Nov/Dec 1987, and the /usr/group UNIX Resources Guide.

If you have a UNIX related event that you wish to publicise then contact either John Quarterman at *jsq@longway.tic.com*, Alain Williams at *addw@phcomp.co.uk*, Susanne Smith at *sws@calvin.uucp* or Carolyn Carr at *carolyn@usenix.uucp*

giving brief details as you see below.

Abbreviations:

APP	Application Portability Profiles
C	Conference
CT&LA	Conformance Testing & Laboratory Accreditation
S	Symposium
T	Tradeshow
U	UNIX
UG	User Group
W	Workshop

mon days	1990 conference	location
Sept 3-7	DECUS Europe Symposium	Cannes, France
Sept 4-6	GUUG C	Wiesbaden, Germany
Sept 11-13	UNIX Service Conference	University of Lancaster, UK
Sept 10-12	UNIX Based Applications, TRUUG C	Istanbul, Turkey
Sept 20-21	Sun UK UG C	Edinburgh, UK
Sept 25-28	AUUG Conference	Southern Cross, Melbourne, Australia
Oct 3-5	Internat'l S of MHS	IFIP WG 6.5, Zurich, Switzerland
Oct 3-5	UNIX Solutions T	Anaheim, California, USA
Oct 4-5	USENIX W -MACH	Burlington, Vermont, USA
Oct 8-12	InterOp 90 ACE	San Jose, CA, USA
Oct 15-19	IEEE 1003	Seattle, WA, USA
Oct 15-19	UUGA C	Vienna, Austria
Oct 17-19	Large Installation Sys. Admin.	Colorado Springs, CO, USA
Oct 22-25	IPA C	Jerusalem, Israel
Oct 22-26	EUUG C	Nice, France
Oct 23-26	ISO/IEC SC22 WG15	Seattle, WA, USA
Oct 25-26	NCR Unix User Group C	Colombia, SC, USA
Oct 28-30	NCR Unix User Group C	Raleigh, SC, USA
Oct 29-Nov 2	SUUG	SUUG & MCNTI, Moscow, U.S.S.R.
Oct 31-Nov 1	UNIX EXPO	New York, NY, USA
Nov 1	NLUUG C	Open Systems, Ede, Netherlands
Nov 5-9	10th Internat'l C on CC	ICCC, New Delhi, India

Nov 7-9	"Unix al Castello" C i2u	Naples, Italy
Nov 12-14	European X Window C, CEP Consultants	Edinburgh, Scotland
Nov 14-16	UNIX EXPO '90 UniForum	Stockholm, Sweden
Nov 15	POSIX APP/OSE Users Forum W	NIST, G, MD, USA
Nov 15-16	16th JUS Symposium	Osaka, Japan
Dec 2-5	SunUG CT	San Jose, USA
Dec 4-5	JUS UNIX Fair '90	Tokyo, Japan
Dec 4-7	IETF	IAB, U. Colorado, Boulder, CO, USA
Dec 10-12	Sinix C, Unix Asia '90	Singapore
Dec 10-14	DECUS S	Las Vegas, NV, USA
Dec 13-16	Sinix T, Unix Pavillion '90	Singapore
Dec 17-19	UKUUG C	Cambridge, UK
	1991	
Jan 7-11	IEEE 1003	New Orleans, LA, USA
Jan 16-18	USENIX, Software Devel. Environments	Grand Kempinski, Dallas, TX, USA
Jan 21-24	UniForum	Infomart, Dallas, TX, USA
Jan 21-25	USENIX	Grand Kempinski, Dallas, TX, USA
Jan 16-17	Multi-User C Show for Gov't UniForum Canada	Ottawa, ON, Canada
Feb 18-22	DECUS S	Ottawa, Canada
Mar	IETF	IAB, Wash. U, St. Louis, MO, USA (tentative)
Mar 13-20	CeBIT 91	Hannover, Germany
Mar 26-30	AFUU C	CNET Paris La Defense, France
April 10-12	IEEE 1003, USENIX, Uniforum, EUUG	Miami, FL, USA (Tentative)
Apr 22-26	DECUS Muenchen Symposium	Hannover, West-Germany
Apr	NCR Unix User Group C	San Antonio, Texas, USA
May	U 8x/etc C&T	/usr/group/cdn; Toronto, ON, Canada
May 6-10	DECUS S	Atlanta, GA, USA
May 9	APP/OSE Users Forum	NIST, G, MD, USA
May 15-17	Multi-User C Show	UniForum Canada, Toronto, ON, USA
May 20-24	EUUG	Tromso, Norway
Jun/Jul	UKUUG C	Liverpool, UK
Jun 10-14	USENIX	Opryland, Nashville, TN, USA
Jun 17-19	Sun User Group	Atlanta, GA, USA
Jul 8-12	IEEE 1003	Santa Clara, CA, USA (location tentative)
Sept 10-12	European Sun User Group CT	NEC, Birmingham, UK
Sept 16-20	EUUG	Budapest, Hungary
Oct 10-11	Multi-User C Show	UniForum Canada, Montreal, Quebec
Oct 21-25	IEEE 1003	Southern Europe (location tentative)
Nov 14	APP/OSE Users Forum	NIST, G, MD, USA
Dec	UKUUG C	Edinburgh, UK
Dec 8-11	Sun Users Group	San Jose Fairmont, USA
Dec 9-11	Sun User Group	San Jose, CA, USA
Dec 9-13	DECUS S	Anaheim, CA, USA
	1992	
Jan 13-17	IEEE 1003	Orlando, FL, USA (location tentative)
Jan 20-24	USENIX	Hilton Square, San Francisco, CA, USA
Jan 20-23	UniForum	Moscone Center, San Francisco, CA, USA
Spring	EUUG	Jersey, UK
Mar 11-18	CeBIT 92	Hannover, Germany

Apr 20-24	IEEE 1003	Montreal, PQ, Canada (location tentative)
May 4-8	DECUS S	Atlanta, GA, USA
Jun 8-12	USENIX	Marriott, San Antonio, TX, USA
Jun 22-24	Sun Users Group	Washington (DC), USA
Jul 13-17	IEEE 1003	Alaska, USA (location tentative)
Autumn	EUUG	Amsterdam, Netherlands
Oct 19-23	IEEE 1003	Scottsdale, AZ, USA (location tentative)
	1993	
Jan	USENIX	Town & Country, San Diego, CA, USA
Mar 15-18	UniForum	Moscone Center, San Francisco, CA, USA
Mar 24-31	CeBIT 93	Hannover, Germany
Jun 21-25	USENIX	Cincinnati, OH, USA
	1994	
Feb 7-10	UniForum	Dallas Convention Center, Dallas, TX, USA
Mar 16-23	CeBIT 94	Hannover, Germany
Jun	USENIX	Boston, MA, USA
Mar 6-9	UniForum	Dallas Convention Center, Dallas, TX, USA
Mar 11-14	UniForum	Moscone Center, San Francisco, CA, USA
Mar 10-13	UniForum	Moscone Center, San Francisco, CA, USA

Organising Bodies

NIST/NBS/POSIX
Roger Martin
National Institute of Standards
and Technology
Technology Building, Room B266
Gaithersburg, MD 20899
+1-301-975-3295
+1-301-975-3295
rmartin@swe.icst.nbs.gov

IEEE Computer Society
P.O. Box 80452
Worldway Postal Center
Los Angeles, Ca. 90080

UniForum (was /usr/group)
2901 Tasman Drive
Suite 201
Santa Clara CA 95054
+1 408 986 8840
+1 408 986 1645 fax

/usr/group/cdn
241 Gamma St.
Etobicoke, Ontario M8W 4G7
Canada
+1-416-259-8122

Tracy MacIntyre
Exhibition Manager
EMAP International Exhibitions Ltd.
Abbot's Court
34 Farringdon Lane
London EC1R 3AU
United Kingdom
+44-1-404-4844

AUUG
P.O. Box 366
Kensington
N.S.W. 2033
Australia
uunet!munnari!auug
auug@munnari.oz.au
+61 3 344 5225

AMIX, c/o IPA
P.O. Box 919
Ramat-Gan
Israel, 52109
+972-3-715770
+972-3-715772
amix@bimacs.bitnet
amix@bimacs.biu.ac.il

Japan UNIX Society (JUS)
#505 Towa-Hanzomon Corp. Bldg.
2-12 Hayabusa-cho
Chiyoda-ku, Tokyo 102
Japan
bod%jus.junet@uunet.uu.net
+81-3-234-5058

UNIX Fair '88 Association
1-1-1 Hirakawa-chu,
Chiyoda-ku, Tokyo 102
Japan

Singapore Unix Association - Sinix
20 Bideford Road #11-05
Wellington Building
Singapore 0922
+65 734 3256

DECUS U.S. Chapter
219 Boston Post Road, BP02
Marlboro, Massachusetts 01752-1850
U.S.A.
+1-617-480-3418

DECUS Europe
1-3, chemin Anneville, Box 176
CH-1213 Petit-Lancy 1
Switzerland
tel: +41 - 22 - 709 42 64
fax: +41 - 22 - 792 25 03

DECUS Munich (for Germany, Austria, Hungary):
DECUS Muenchen e.V.
Freischuetzstr. 91
D-8000 Muenchen 81
Germany
tel: +49 - 89 - 95 91 - 44 30

USENIX Association Office
2560 Ninth St., Suite 215
Berkeley, CA 94710
USA
+1 415 528 8649
office@usenix.uucp

National Expositions Co., Inc. (UNIX EXPO)
15 West 39th Street
New York, NY 10018
U.S.A.
+1-212-391-9111
fax: +1-212-819-0755

USING
P.O. Box 1077
Lisle,
Illinois 60532, USA

Sun UK User Group
Sue Crozier
Sun Microsystems, UK
+44 276 20980

Sun User Group, Inc.
Peter H. Salus
PO Box 167
Cambridge, MA 02142
USA
+1 617 739-0202
peter@uunet.uu.net

UniForum NZ Secretariat
PO Box 585
Hamilton
New Zealand

TRUUG
Esref ADALI,
Professor of Control and Computer Engineering,
TRUUG UNIX '90 Chairman,
Istanbul Technical University
Ayazaga
Istanbul
Turkey
+90 1 176 3586

EUUG National group addresses can be found on the back cover of this newsletter.

EUUG Spring Conference Abstracts

Here are the abstracts of the papers the were delivered at the EUUG spring conference held in Munich. Copies of the proceedings may be obtained from the EUUG at Owles Hall at a cost of £20.

Thanks are due to Stuart McRobert <sm@ic.doc.ac.uk> who typeset the proceedings and provided the abstracts.

The Development of an Internet Protocol Routing Gateway

Richard Almeida

The Computing Laboratory
The University
Canterbury, Kent CT2 7NF
England
rpa@ukc.ac.uk

The Design of a PC-based NFS Client from an MS-DOS Programmer's Perspective

Ian Chapman

Siemens plc
Systems Development Group
65-73 Crockhamwell Road
Woodley, Reading
Berkshire, RG5 3JP
England
+44 734 443 042
ian@siesoft.co.uk

Siemens Distributed File System (DFS) is a client-only NFS implementation for IBM PCs and compatibles. The project was undertaken because a similar product did not exist to run with Excelan Ethernet hardware. The Project Team was composed primarily of MS-DOS programmers, which led to a number of interesting differences in the design of DFS compared to the other PC-based NFS clients on the market. This paper gives an overview of Siemens DFS, highlighting these differences.

Archiving of Documents on WORM Disks – the NFS Approach

Helmut Kalb
Snoopy Schmitz

iXOS Software GmbH
Bretonischer Ring 12
8011 Grasbrunn/Munich
West Germany

We would like to present some work we have done in a contract for SIEMENS, Germany. The project was to allow UNIX users to store NCI documents on optical disks. Due to legal reasons WORM disks were chosen because the immutability of the medium makes them ideal media for archiving. Our solution is a client-server implementation that has the following features

On the server side:
- 4 WORM-disks, jukeboxes and printers are connected via SCSI-2 interfaces
- UNIX and NFS transparent file system
- no changes in kernel
- autonomous information on the WORM-disks
- unique identifiers for all objects
- transactions
- additional SQL interface to the archive
- administration of on-line and off-line archives

and on the client side:
- 4 Fax group IV compression for NCI documents
- display via X-Windows and MOTIF
- special client programs for mass scanning support
- several different client programs for the archive administration

Breaking Through the NFS Performance Barrier

Joseph Moran
Russel Sandberg
Don Coleman
Jonathan Kepecs
Bob Lyon
jkepecs@legato.com

Legato Systems, Inc.
260 Sheridan Ave.
Palo Alto, CA. 94306
U.S.A.

The Sun Network File System (NFS) has become popular because it allows users to transparently access files across heterogeneous networks. NFS supports a spectrum of network topologies, from small, simple and homogeneous, to large, complex, multi-vendor networks. Given the diversity and complexity of NFS environments, isolating performance problems can be difficult. This paper identifies common NFS performance problems, and recommends specific practical solutions. In particular, we evaluate the impact of reliable write caching on NFS performance.

HIT-Multicode – Implementation of a multilingual Word Processing System

Thomas Merz

InterFace Connection GmbH
Weißenburger Str. 43
8000 München 80
West Germany

Current internationalisation concepts are suffering from a major disadvantage: they are limited to one particular codeset. This codeset may be designed according to the needs of a specific language, even based on a non-Latin alphabet such as Greek or Arabic. However, the requirement for mixing of different languages and scripts which frequently arises in modern communications, cannot be met by such systems. The only way to cope with this problem efficiently is to use multicode techniques, i.e. software which abandons the restriction of a single codeset.

An X-windows Teletext Service for UNIX

George M. Taylor
Jim Reid

Department of Computer Science,
Strathclyde University,
Glasgow,
Scotland,
G1 1XH.
gtaylor@cs.strath.ac.uk
jim@cs.strath.ac.uk

Teletext information in the UK has been available on all four of the national television channels for some time now. The aim of this project is to provide the Computer Science Department with a network-wide, UNIX-based service for viewing the broadcast teletext information. This will use the X window system to present the information on a bit-mapped workstation display. An alternative display for dumb terminals will also be developed as part of this project.

In this paper we describe a final year student project to design and implement such a service. The work entails the design and construction of a decoder and interfacing it to a UNIX system, development of system software – a device driver, daemons and client/server processes – and a user interface. The paper presents an overview of the project and the problems encountered in its development.

Porting Banking Software from VMS to UNIX

Rudolf Koster

InterFace Computer GmbH
Garmischer Str. 4
D-8000 München 2

This paper describes the process of porting and partially re-implementing one of the first UNIX software packages of substantial size for the banking sector. The *Deutsche Terminbörse* began operation in January 1990 and consists of networked minicomputers running VMS or AIX, IBM's version of UNIX. During the port to AIX, special care had to be taken to strictly retain functionality and to guarantee fairness to traders.

Is Unix resistant to computer viruses?

Pascal Beyls

BULL
1, rue de Provence
Echirolles
FRANCE
beyls@ec.bull.fr

Epidemics always catch people's imagination. It was the case with the plague during the Middle Ages. More recently, computer viruses have appeared, and some of them, like Friday the 13th have attracted considerable media attention. The means of contamination (in cauda venenum), their reproduction and their spread show real similarities with biological viruses. In the same way, they are a scourge for all users.

The distribution of applications in binary format and intensive use of networks speed up the spread of these viruses, aided even further by the ignorance, naivety and incredulity of data processing managers. Talk about viruses and the danger they represent causes amusement, derisive smiles, condescension and even compassion. But it does not just happen to other people.

To fight such an adversary, we have to get to know it. This is why it can be useful to define the virus, describe the inoculation method, explain its workings, its spread and even its mutations.

In this field, the authors show great imagination. As well as simple destruction of files, the viruses poison† users' lives: ping-pong balls on the screen, little gnome closing all the windows on the screen, indelible message on printers, etc.

The user is not entirely defenceless. Each virus can be identified and neutralised using programs known as vaccines and disinfectants. But, as is the rule during epidemics, prevention remains better than a cure. This is particularly true as it is always difficult to get rid of viruses.

Both micros and mainframes are ideal environments for virus proliferation. Nevertheless, there are few examples mentioned where UNIX systems were contaminated. Is UNIX resistant? Do its mechanisms provide it with defences? Or is it simply because it has not been possible to isolate one?

The aim of this paper is to make a brief survey of the situation.

The Administerability of UNIX Systems

Norbert Ondra

Department of Computer Science
University of Klagenfurt
Austria

Today, UNIX enjoys a wide-spread use all over the university world, both in research and education areas. The main reasons for this are the high degree of availability, portability, compatibility and its tailorability towards the individual needs of specific application areas. One important issue, however, is often left out of consideration when talking about the virtues of UNIX, namely administerability.

While this issue might be a minor concern in small experimental environments, it becomes most relevant in larger and more complex network environments, in particular when reliable production operability must be safely and easily provided. Unfortunately, UNIX systems show substantial shortcomings in both enabling, facilitating and supporting system administration, thus handicapping the goals of system availability, security and integrity to be achieved. As will be shown in typical examples, the deficiencies cover the whole range from conceptual issues through insufficient tools up to incompatibilities between UNIX systems, making system administration a tough and time-consuming job.

Hence, bad administerability compromises the requirements of production environments, e.g. in commercial or business areas, where those main goals must be easily attainable with negligible effort. As a consequence, it will turn out to be necessary to significantly improve the administerability of UNIX systems through enhanced concepts currently missing even at kernel level and through providing a standard toolset for system administration routine.

In the opinion of the author, this is a serious challenge for the UNIX manufacturers' community to come up with really

0. † The word virus comes from the Latin for poison.

administerable UNICes which comply with the still evolving standards and all the requirements of system administration in complex, networked (production) environments, where administerability will become a major decision making factor.

Authentication Using a Chip Card

Stanislaus Gefroerer
Ingo M. Hoffmann

Siemens AG
Otto-Hahn-Ring 6
8000 Muenchen 83
Germany

To protect information effectively against unauthorized access it is essential to have fail-safe access authorization identification. As is well known, password procedures have their limitations in this respect, especially when used in computer networks. Today, chip card technology provides a means of solving this problem. It enables chip card owners to identify themselves definitively as the rightful owner of the chip card, and thus to prove their identities. A similar proof can be conducted by the computer vis-a-vis the chip card owners. This technique is used for system access to SINIX systems† and is also provided as a program interface for ordinary application programs. The corresponding software product with the associated hardware is available under the name ASECO ‡.

Architecture of a Collaborative System Using Smalltalk and Unix

C Vieville
A Derycke
P Vilers

Trigone Laboratory
University of Lille
Vieville@frcitl71.bitnet

Our team of Trigone laboratory works in the field of "New Technologies of Education". The aim of this work is to develop tools which can help tutors and students. The main features of these tools include an user-friendly interface as well as important communication facilities between the users' workstations. The paper will show the various problems that we have had to work out so as to fit Smalltalk to UNIX in order to set up a rapid prototyping platform.

After an explanation about the choice of Smalltalk, we will further specify our communication needs before analysing Smalltalk's handling facilities of external events. We shall end with a description of our communication architecture upon the sockets.

ITHACA: An Overview

Anna-Kristin Pröfrock†
Martin Ader
Gerhard Müller
Dennis Tsichritzis

Nixdorf Microprocessor Engineering GmbH, Berlin

0. † SINIX is the Siemens version of UNIX.

0. ‡ Advanced SEcurity COncept.

† The material contained in this paper was produced by the respective working groups.

This paper describes the Ithaca project (Integrated Toolkit for Highly Advanced Computer Applications) developed under ESPRIT contract 2121 as a technical integration project. Ithaca is an integrated application support system which is evaluated by demonstrator applications, even at an early stage of development. To a high degree, the Ithaca environment supports an object-oriented approach to application development. Thus, it consists of a kernel including a concurrent object-oriented programming language called CooL, with support of persistent objects through integration of a structurally object-oriented database system interface called NooDLE. Development of object-oriented applications is supported by tools which build an application development environment. Here, special emphasis is placed on a modified object-oriented life cycle in application development. The users of an Ithaca application will be supported by a user support system based on X-Toolkit. The Ithaca office workbench includes a multi-media environment and supports typical office organisations. Other demonstrators to evaluate Ithaca are a chemistry workbench, a financial workbench and a public administration workbench.

Clean Semantics of Multiple Inheritance

N Giambiasi
C Oussalah

Laboratoire d'Etude et Recherche en Informatique
Parc Scientifique Georges Besse
30000 Nmes - France
L Torres
ITECA
Centre ATRIA - 5, Bd de Prague
30000 Nmes - France

The ORL† elaboration led us to propose an inheritance model based on the definition of various graphs. Two versions of the language concern knowledge without conflict, called exact knowledge, and knowledge where conflicts may appear or inexact knowledge.

BUD – Backtrackable UNIX Data Control

Zs. Hernath
M. Szokolov

Computer Centre
Eötvös Lorahd University
Budapest
Bogdanfy u. 10/B
H-1117 Hungary

BUD is a C-interface resulting from research in database management that has been carried out at the Computer Centre of Eötvös Loránd University since 1986. Our original goal was to create a user-friendly low-level interface for developing higher level file systems to be used in a multi-user environment with transaction-oriented concurrency control. The interface was designed and implemented as a minimal extension of the low-level I/O and memory management facilities of standard System V. To test the system and to demonstrate concurrent processes running under BUD control, the BUD functions were also implemented as executable commands that can be activated from the menu of an interactive DEMO system. BUD

† ORL: Object Representation Language, object oriented language developped in LERI, inspired by frames.

is not an interface for the end user. It is a powerful tool for system and application programmers who implement data systems.

The Educational On-Line System

Nick Williams

Imperial College
University of London
UK
njw@doc.ic.ac.uk
William Cattey
Robert French
Bruce Lewis
Massachusetts Institute of Technology
USA
wdc@athena.mit.edu
rfrench@athena.mit.edu
brlewis@athena.mit.edu

The Educational On-line System (EOS) is a suite of programs which enables students and lecturers to exchange course materials within the Project Athena environment at the Massachusetts Institute of Technology in Cambridge, MA, USA. Early versions of this software used a command-line interface. This paper outlines the implementation of a user interface to the file exchange system, using the Andrew Toolkit (ATK) to produce an X Window System application.

Dynamic Driver Loading for UNIX System V

Dieter Konnerth
Elmar Bartel
Oliver Adler

Idev Software, Munich
dieter@idefix.uucp

Dynamic Driver Loading (DDL) is a concept which makes it possible to add, remove and replace driver modules from the running UNIX System at any time. The paper presents this concept, related concepts and the design goals of our implementation for System V.3.2 386. Details of the implementation are explained. In the second part of the paper we focus on problems which we met in the course of implementation and use of the system. The problems appear to be partly due to the very basic design of the kernel/driver interface, but most are due to breaking the rules of this interface. We conclude that the system is an invaluable tool for driver development, that its usage forces developers to write clean drivers, which improves the quality and portability of driver code, and that the concept is usable in production systems if the drivers are carefully designed.

The XView User Interface Toolkit Object Model

Nayeem Islam

Sun Microsystems
2550 Garcia Ave.
Mountain View CA 94043
islam@sun.com

This paper describes the object-oriented programming model in XView, a user interface toolkit written in C and based on the X11 window system. The mechanisms by which XView supports two important aspects of object-oriented programming, data abstraction and inheritance, are discussed. This paper describes how the object-oriented programming model in XView maps onto the window systems environment. This approach is compared with the another object-oriented toolkit implemented in C, the *Xt Intrinsics* from the MIT X consortium.

Interface between UNIX and HyperCard

Mikael Wedlin

Dept of Computer and Information Science
Linkoping University
mwe@ida.liu.se

This work describes a method of using the benefits of UNIX as a developing system and targeting the result to, in my opinion, a more user friendly system, HyperCard on a Macintosh. HyperCard can best be described as a programmable information handler.

UNIX is an operating system written by programmers, for programmers and few ordinary users will ever have any advantage of the complex design of UNIX. They often use some simpler personal computer instead, often with a window based interface instead. These systems are often difficult to program due to the lack of standardisation and a simpler operating system. Window systems are also rather difficult to program due to their complex interaction with the user.

My goal was accomplished by writing a program that could convert a UNIX program to a code resource that was linked to a HyperCard stack. This program then works as an external command (XCMD) in HyperCard.

Weaknesses in Shared Memory and Software Interrupts in UNIX

D L Jenkins
R R Martin

University of Wales College of Cardiff
Department of Computing Mathematics
PO Box 916
Cardiff
CF2 4YN
Wales
United Kingdom

djl@cm.cf.ac.uk
ralph@cm.cf.ac.uk

The analysis of a sketch requires an application capable of recognising basic graphics primitives, for example lines or circles, and determining geometric relations that might exist between them. However, sketching is highly interactive requiring any application to be very responsive during input whilst the sketch analysis is given a lower priority. This paper presents a brief description of a sketch analysis program called Easel, an overview of possible architectures, UNIX mechanisms to support them, and a detailed discussion of two implementations. In addition, the problems associated with supporting these models with UNIX is then outlined.

INGRID: A Graphical Tool for User Interface Construction

L Carriço
N Guimarães
P Antunes

INESC
Rua Alves Redol, 9, 6D
1000 Lisboa
Portugal

lmc@inesc.inesc.pt
nmg@inesc.inesc.pt

paa@sabrina.inesc.pt

INGRID is an interactive tool for user interface construction. The tool enforces a specific user interface model that considers both the functional composition of the user interface elements and an object-oriented approach as the fundamental design and development methodology.

The implementation of INGRID highlights three main components: a run-time support system for interactive programming (in), a toolkit that defines the abstract interfaces to the several components of the UI allowing integration of multiple graphical toolkits, and the user interface of INGRID itself.

Advances with the MACH Operating System

Richard F. Rashid

Carnegie-Mellon University

This paper was not submitted in time for inclusion in the proceedings.

Multi-threaded Processes in CHORUS/MIX

François Armand
Frédéric Herrmann
Jim Lipkis
Marc Rozier

Chorus Systèmes
6, avenue Gustave Eiffel, F-78182,
Saint-Quentin-En-Yvelines, France
Tel: +33 1 30 57 00 22,
Fax: +33 1 30 57 00 66,

mg@chorus.fr

Interest in concurrent programming in recent years has spurred development of "threads", or "lightweight processes", as an operating system paradigm. UNIX-based systems have been especially affected by this trend because the smallest unit of CPU scheduling in UNIX, the process, is a rich and expensive software entity with a private memory address space. In this article we examine performance constraints affecting concurrent programs, including real-time applications, in order to understand and evaluate the demand for a new scheduling model. Although performance criteria differ sharply among various application domains, we conclude that a single thread model can provide efficient concurrent execution in a general-purpose operating system.

We describe the design considerations behind the thread-management facilities of CHORUS/MIX, a UNIX-compatible operating system built for distributed, real-time, and parallel computing. Mechanisms for processor scheduling and inter-thread synchronization must satisfy the needs of each of these three categories of concurrency. Extension of the traditional UNIX interface to the multi-threaded environment is an area of particular delicacy. CHORUS/MIX adopts novel approaches for signal handling and other UNIX facilities so as to ensure a smooth transition from sequential to concurrent semantics in applications.

Distributed Computing in Heterogeneous Environments

Herman Moons,
Pierre Verbaeten

Katholieke Universiteit Leuven
Dept. of Computer Science
Leuven, Belgium

herman@cs.kuleuven.ac.be

Ulf Hollberg
IBM European Networking Center
Heidelberg, Germany

Distributed computing systems have received considerable attention in the last decade. Unfortunately, current research efforts are often restricted to homogeneous environments. There seems to be little attention for real-world installations, where heterogeneity is mostly the rule, rather than the exception.

The *Distributed Academic Computing Network Operating System*, DACNOS for short, presents a general solution for running distributed applications in heterogeneous networks. The DACNOS extends local guest operating system services to provide homogeneous networking functionality. This is achieved by a virtual global object space, to which each user has access from within his native environment.

This paper describes the architecture of the DACNOS network operating system, with special emphasis on its incarnation on the UNIX platform. It presents the DACNOS solutions to the problems of communication, access protection and data representation in a network of cooperating heterogeneous systems. DACNOS implementations currently exist for VM/CMS, VAX/VMS, PC-DOS, OS/2 and UNIX System V derivatives.

AIX Version 3 on the RISC System/6000 Family

Iain Elliot

IBM Deutschland GmbH

The Advanced Interactive Executive (AIX) Version 3 UNIX kernel was designed to be portable across a wide range of architectures from the Intel 80x86 microprocessor family up to the IBM System/370-XA architecture mainframes. In several areas, however, the AIX Version 3 software was optimised to make full use of the RISC System/6000 architecture features; in particular, the virtual memory management, the device handlers and the compiler code generators. This paper reviews the virtual memory management subsystem and the radical alterations made to other parts of the kernel to support it. These concepts may be considered as the key to understanding the AIX Version 3 kernel on the RISC System/6000 computers.

The memory management hardware is described in detail, the implications of the hardware are then examined and the resulting software features are shown. Finally, the benefits for the user are outlined.

Cooperation Support for UNIX-Based Industrial Applications

Holger Herzog
Burkhard Stork

Siemens AG,
ZFE F2 SOF 4
Otto-Hahn-Ring 6,
D-8000 Munich 83,
FRG
sto@ztivax.uucp

RApp: A Generic Routing Application in C++.

Sanjiv Gossain
Bruce Anderson

Department of Electronic Systems Engineering
University of Essex,

Colchester, UK
goss@essex.ac.uk
bruce@essex.ac.uk

A new approach to application development within domains, based upon software reuse, is presented the generic application. A generic routing application, RApp, is used as a vehicle to outline the features of the generic application. RApp's constituent classes are described and the method by which applications are created is outlined. Using object-oriented techniques, we have provided flexibility, maintainability and reusability, in the form of a class hierarchy achieving up to 76% reuse of code. We have also reused designs from previous implementations. The significance of these and other results are presented.

PatMat — a Pattern Matching Class

Peter Polkinghorne

GEC Hirst Research Centre
East Lane
Wembley
Middlesex HA9 7PP.
UK
pjmp@gec-rl-hrc.co.uk

The paper describes the process of using an Object Oriented language to design and implement a pattern matching system for operating on text. The pattern matching system is implemented as a *class* called *patmat*. The design is based on *Snobol4*. The object oriented facilities provided by both provide a good user interface and a mechanism for the implementation. Details of the development statistics are given along with the tools used. The performance of a *patmat* based *grep* program is compared with a variety of others. Finally a detailed description of the class interface is given.

Network Management Gateways

Gabriele Cressman-Hirl

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
(415) 336-1085
gabi@eng.sun.com

The effectiveness of managing large and complex multi-vendor networks depends on the availability of network management platforms that specifically address the diversity of heterogeneous networks. This paper discusses design requirements, architecture models and implementation issues for managing different types of networks simultaneously. The theoretical discussion is illustrated by examples of a case-study conducted to integrate two different network management protocols and network architectures.

An introduction to OSI architecture, concepts and terminology

John Henshall

EUCS
University of Edinburgh
Edinburgh
Scotland
J.Henshall@ed.ac.uk

This paper will take the form of a tutorial slide presentation as an introduction to the application orientated layers of the ISO reference model. It is aimed at the level of no knowledge of OSI and goes from there to a point where the listener should have a reasonable grasp of the basics of design and terminology of OSI. It is not a "justifying" paper – neither knocking nor glorifying any particular communications technique – it is a simple tutorial. There are 30+ slides hopefully providing an ideal "starter" for OSI communications literacy both for those who wish to join the ranks of its supporters, or to have more buzz words to use in the fight against its rather "top heavy" design.

PCSERVE: An Attempt To Integrate PC Users Into The UNIX Community

Toshiharu Harada
Takehiko Nishiyama
Hidekazu Enjo

NTT Data Communications Systems Corporation
Kowa Kawasaki Nishi-guchi Bldg.,
66–2 Horikawa-cho, Saiwai-ku, Kawasaki-shi,
Kanagawa 210, Japan
harada@rd.nttdata.jp

PCSERVE is a system which integrates PC (hereafter, PC stands for a DOS based Personal Computer) users into the UNIX environment. Using PCSERVE, PC users are able to access the UNIX community without a knowledge of UNIX.

The system is based on the server-agent-client model, which is an enhanced version of the server-client model, by introducing an agent between a client and a server. The agent is placed on a UNIX host and provides methods to invoke and utilize anonymous stream type internet services in UNIX hosts according to the requests from sophisticated applications on PC's.

This paper describes the design issues and their consequences. With the introduction of the agent, PCSERVE achieved a higher level of flexibility and solved the existing problems of the server-client model between PC's and UNIX hosts.

NLUUG Spring Conference Abstracts

The NLUUG had its spring conference on 17 May 1990 with the theme "UNIX & Parallelism". The conference proceedings were not ready in time for the conference, but were sent to the 180 participants later.

Copies of the proceedings can be ordered by EUUG members from the NLUUG (their address is on the back cover). The price is Hfl. 40, (incl. postage). The proceedings are predominantly in Dutch.

Below you find the (translated) abstracts of the 13 presentations:

An experimental approach towards Parallel Computing

L.O. Hertzberger

Universiteit van Amsterdam, FWI
Kruislaan, 1098 SJ Amsterdam

Parallel computing has not lived up to the expectations that were generated by large computer projects of which the Japanese Fifth Generation Computer System project has attracted the largest interest. This does not imply that parallel computing is not a promising direction to go for the future. However, the subject turned out to be a more difficult research topic than was believed in the past. Like any other topic in research it requires a systematic approach of which experimentation with parallelism is a methodology that can provide answers for a number of not well understood issues like; optimal granularity, load balancing, the problem of exploitation of locality, etc.

In this presentation a number of aspects of parallelism were discussed. What is necessary to design an experimental facility will be illustrated. In this discussion the role of the operating system will also be taken into account.

Procol, a parallel object-orientated language

Chris Laffra

RU Leiden, Wiskunde & Informatica
Postbus 9512, 2300 RA Leiden

Procol is an object-orientated C-extension with delegation. Procol stimulates concurrency: objects execute parallel if they don't communicate. An object produces a set of implicated operations. Communication uses remote procedure calls, or uses direction messages (with short bindings). In order to communicate the sender and the receiver have to be specified, either with object variables or with an object type. For this the following client-server combinations are possible 1-1, n-1 or 1-n.

Procol controls the excess to an object with an explicit protocol. This protocol controls the permission and the successive interaction between an object and its clients. The use of protocols produces a first step to a structured, save and potential verifiable exchange of information between objects.

Procol's communication binding is dynamic (run-time). Because of this Procol basically operates in a distributed, dynamic and incremental object environment.

Parallel Factoring by Electronic Mail

Herman te Riele and Walter Lioen

Centrum Wiskunde & Informatica
Kruislaan 413, 1098 SJ Amsterdam

The multiple-polynomial quadratic sieve is the fastest known general purpose factoring algorithm. A.K. Lenstra and M.S. Manasse have organised a world-wide project for the parallel execution of this algorithm. Electronic mail is being used for the distribution of the program and for the interprocessor communication. Anybody with e-mail facilities, a C-compiler and idle computer cycles (and the possibility to run a 1.2 MByte program and store a 1 MByte file) can participate in this project.

Numbers of up to 106 decimal digits have been factorised so far in this project. CWI has contributed (otherwise idle) computer cycles from SUN 3 workstations, SPARCstation 1's and an Alliant FX/4 parallel processor. In this talk, we will describe our experiences in this project. Background for this project is the following question, interesting for public-key cryptography: how big are the integers we can factor with our present algorithms?

Parallel Programming on Unix Multiprocessor Systems

P.A. Lee

Centre For Multiprocessors
Computing Laboratory
University of Newcastle upon Tyne, U.K.

Parallel programming on any system requires three key features: 1- multiple threads of control, 2- data communications for shared information, 3- synchronisation mechanisms.

These features can be provided at various levels in a multiprocessor: some are supported by the hardware, some are implemented by the operating system, and others are provided at the user-level. Moreover, the cost of mechanisms is of vital importance to parallel programming. This talk will examine the means by which these features are implemented on the Encore Multimax multiprocessor system, and will present some of the results that we have obtained in using these features to parallelise programs.

Parallel Compiler Technology

J. Penn

Alliant Computer b.v.
Weverstede 1, 3431 JS Nieuwegein

Mr Penn discussed in detail the automatic parallelising compilers, developed by Alliant, in particular for Intel's i860 RISC processor architecture, for instruction scheduling, model for concurrency and vector transformation, automatic cache blocking and RISC vectorisation.

Concurrent Functional Programming

M.J. Plasmeijer, M.J.C.D. van Eekelen

KU Nijmegen, dept. Computer Science
Toernooiveld 1, 6525 ED Nijmegen

Functional languages have as advantages that they have a high expressive power and that correctness is relatively easy to achieve. An important property of functional programs is that expressions can be evaluated in any order, albeit that some reduction orders may lead to non-termination. If a result is obtained, it will always be the same independent of the chosen evaluation order. This property makes functional languages very suited for interleaved and parallel evaluation.

With a couple of simple primitives the potential power and elegance of concurrent functional programming is demonstrated. In a concurrent functional language processes are functions that are executed concurrently. By using mutual recursion arbitrary dependencies between these functions can be specified thus creating a way to define arbitrary networks of processes. The communication and synchronisation between the processes can be realised with the lazy evaluation principle. No special communication primitives are needed: communication takes place when a process demands a value that is being calculated by another process.

Parallelism: an unfulfilled promise

Albert van der Horst

BSO
Postbus 8052, 3053RB Utrecht

There are quite a lot of theoretical essays about the possibilities and restrictions of computers, sequential computers, analogue computers, etc.

This is not always similar to the experiences users have, and even sometimes in complete contradiction with the facts. In this presentation we will look closer at some of these hot issues. Using examples of how UNIX and a simulation environment (MANIP) handle things, we will demonstrate that the problems that occur have enough parallelism them in order to use parallel computers in a meaningful way.

Neural Networking

Hans de Hartog

Digital Equipment bv
Europalaan 44, 3526 KS Utrecht

An update on the research of neural networks

In its R&D department Digital is working on Neural networks. A preview, state of the art of the simulation (pre-processor for C), and the practical useful results (including integration in existing software packages).

Commercial Parallel UNIX in real life

Ir. M.J. Mathijsse

Sequent Europe B.V.
Locatellikade 1, 1076 AZ Amsterdam

Sequent is a supplier of the new generation parallel symmetric supermini computers. The concept of the hardware architecture is based on a central bus with several attached resources in the shape of processors, memory and peripherals. The Symmetry family, the present system of Sequent, is based on the Intel 80386 micro-processor, a system which can have 2 to 30 processors.

All processors work simultaneously on shared memory, in which each processor has its own 128 Kbytes memory-cache. The cache-technology is based on 'two way set associative' cache memories, which are implemented with the advanced 'copy-back' methodology.

The Operating System Dynix/PTX is fully compatible with UNIX System V.4 and is fully aimed at a parallel symmetric computer architecture.

Parallelism of UNIX has influence on the multi-processing aspects of the Operating Systems as well as the possibilities for the programmer to parallelise applications (parallel processing).

Sequent has implemented next to the multi-processing aspects of UNIX a number of unique methods for run-queue, load distribution- and I/O- organisation.

Because of the qualities of the hardware architecture the programmer or designer can use different programming techniques to parallelise his own applications.

Neural Networks

Louis Vuurpijl

KU Nijmegen, Mathematics and Informatics
Toernooiveld 1, 6525 ED Nijmegen

In recent years there has been an enormous increase of interest in neural networks. In this presentation we will explain what neural networks actually are and for what purposes they are used. Furthermore we will make a comparison between neural networks and parallel computer architecture. Also we will focus on the research project SPIN "neural network project" which is being conducted at the KU in Nijmegen.

The Implementation of C on a Dataflow Graphical Workstation

Arthur H. Veen

Parallel Computing
Postbus 16775, 1001 RG Amsterdam

Dataflow Technology Netherlands is designing a graphical workstation based on 32 dataflow processing elements. The first commercial machine should become available this year. In this presentation we will quickly review the machine, while the major subject will be the RC compiler made by Parallel Computing. In contrast to the usual declarative way of programming dataflow machines, the RC is programmed in an imperative way. (It is a rather limited version of C). Many of the problems that occurred during the implementation of the compiler had to do with the architecture instead of the imperative character of the language.

The operating system Helios

Jonathan Powell

Perihelion Software Ltd.
The Maltings, Charlton Road Shepton Mallet
Sommerset BA4 5QE, U.K.

Helios is an operating system specifically designed for the transputer and was intended right from the start to run on multiple processors. Although appearing similar to UNIX at the user level, the underlying implementation is entirely different in order to handle this.

Helios is based on the client-server model for operating systems, a technique which is widely used in many current systems. This paper describes some of the major features of the underlying system kernel and the tools provided with Helios. Also a brief description of the management of parallel tasks and their specification is presented.

Spirit Workstation

Martijn de Lange

ACE, Associated Computer Experts,
Van Eeghenstraat 100, 1071 GL Amsterdam

Parallelism and Heterogeneity in the SPIRIT Workstation

Distributed technology and multi-processing tend to take less rigid stands in considering a shared memory approach in combination with distributed working today. The high performance SPIRIT Workstation is based on a shared memory multi-processor kernel geared to operate in a heterogeneous hardware system. Aspects such as threads, segment sharing, and both coarse grain and fine grain parallelism are offered and the approach taken to provide for heterogeneity in a transparent way to the user is described. An
overview of layers and flavours in the pertaining parallel OS'es is given while describing the rationale for those facilities as selected to be available in SPIRIT.

OPEN Systems & UNIX is the theme of the NLUUG fall conference November 1, 1990, in "De Reehorst", Ede.

UKUUG Summer Conference Abstracts

Here are the abstracts of the papers the were delivered at the UKUUG conference held in London in July. Copies of the proceedings may be obtained from the UKUUG at Owles Hall at a cost of £50.

Thanks are due to Stuart McRobert <sm@ic.doc.ac.uk> who typeset the proceedings and provided the abstracts.

XEUS: An intelligent terminal system

Laszlo Biczok
Kalman Szeker

Central Research Institute for Physics
P.O. Box 49
Budapest H-1525
Hungary
sztaki!ella!h0707var

XEUS is a UNIX based software environment that makes it possible for PC (DOS) users to work under a more powerful operating system without having to change from their familiar user interface and hardware environment. The XEUS system is based on a host machine running UNIX, surrounded by IBM PCs used as intelligent terminals. A unique terminal emulator program provides an alternative to the terminal user: working under UNIX with the emulator or running DOS and using the UNIX file system of the host computer for background storage. The system provides methods for a UNIX program running on the host machine to download routines to a terminal. Additionally, by offering synchronization and parameter passing facilities, XEUS can support distributed processing.

Developing Document Management Systems Using the Andrew Toolkit

Martin D. Beer

Department of Computer Science,
University of Liverpool,
P.O. Box 147,
Liverpool,
L69 3BX
mdb@mva.cs.liv.ac.uk

This paper will describe some of the work that is currently in progress at Liverpool, with the intention of producing more sophisticated tools for the design, authoring and management of large document packages. Considerable progress has been made by using the Andrew document editor toolkit as the basis for further development. The principle advantage of doing this is that it is no longer necessary to expend considerable effort on the implementation of the basic editing components that are required for any such system. These are all readily available in Andrew.

Until recently, a serious deterrent to following such an approach has been the difficulties encountered when writing programs using the Andrew toolkit. The most recent release does however contain a number of tools that greatly assist in this process. After discussing the use of the Andrew Development Environment Workbench (ADEW), one such tool, progress in developing two sample applications: a lecture slide management package, and a simple notes program, is reviewed.

The development cycle of the Andrew applications is discussed, including the difficulties encountered. In particular the various tools used from within the Andrew toolkit are described, showing how their functionality was used, and the failings, if any, in the their operation during the development cycle of these programs.

G3 – A Language for Typesetting Three Dimensional Graphics

Andrei G. Yaneff
Trevor I. Fenner

Department of Computer Science,
Birkbeck College,
University of London,
Malet Street,
London, WC1E 7HX,
England
ubacw65@cs.bbk.ac.uk

A language for typesetting three-dimensional graphics as an integral part of a document preparation system is presented. g3 can handle a variety of 3D objects such as polylines, polygons, solids of revolution and extrusion, as well as parametrically defined curves and surfaces. Arbitrary 3D views can be specified. The system can be used to typeset 3D illustrations of hierarchically specified graphical objects and for 3D function plots.

Development of a Distributed Revision Control System (DRCS)

Brian O'Donovan
Jane Grimson

Dept. of Computer Science,
Trinity College,
Dublin 2,
Rep. of Ireland.
odonovan@cs.tcd.ie

This paper describes the implementation of a Distributed Revision Control System (DRCS). This system allows users who are geographically dispersed over a number of different sites to transparently share access to multiple versions of files. It is developed as an enhancement to an existing version control system called RCS (Revision Control System) which is already in widespread use on UNIX systems. DRCS runs on a variety of hardware/software platforms and supports a number of different communications protocols.

Large-Scale Software Development Under UNIX

Stuart Feldman

Bellcore
Morristown, NJ, USA
sif@bellcore.com

People insist on developing large pieces of software. This talk will consider some of the issues that arise when big projects are undertaken, and some of the advantages and disadvantages of the use of UNIX as base for such development activities.

The UNIX System and Software Productivity

Brian W. Kernighan
bwk@inet.att.com

Software productivity is a perennial issue, as organizations search for ways to cut the ever-rising cost of producing software.

Although it is not always apparent, we have made true progress – languages, operating systems, tools, and methodology are all much superior to those of a decade or two ago, and as a result programmers can be more productive. In this paper, I will discuss how the evolving UNIX environment has contributed to improved software productivity, and also suggest where to search for further progress.

A High Capacity TCP/IP in Parallel Streams

Ken Dove
Sequent Computer Systems
15450 SW Koll Parkway
Beaverton, OR, 97006-6063
USA
uunet!sequent!kfd

MESHIX:
A UNIX like Operating System for Distributed Machines

Phil Winterbottom
Tim Wilkinson
Department of Computer Science, City University,
Northampton Square, London EC1V 0HB.
philw@cs.city.ac.uk

Topsy is a new message passing multicomputer architecture. It comprises a fully UNIX compatible distributed kernel, called MESHIX, resting on a message based software system which in turn relies on new, high speed, communications hardware. A brief overview of the machine architecture and operating system is presented. We then discuss the implementation of the distributed file servers and their impact on cache policies and the virtual memory system.

Cambridge University Pilot FDDI Network

Tony Morris
Cambridge University Computing Service
Corn Exchange Street
Cambridge
England
CB2 3QG
awml@cl.cam.ac.uk

Cambridge University Computing Service are nearing the end of the planning phase for a Metropolitan Area Network covering all University sites in the city of Cambridge. This network will be an optical fibre network for the interlinking of computer networks but also capable of carrying voice, image and other forms of traffic. We have recently commissioned five Sun 4/390 Sparcservers each with X.25, Ethernet and FDDI interfaces. A small local fibre network interconnects them. These machines will provide a Central UNIX Service to complement the service provided by our IBM 3084 and are the pilot FDDI network.

Beyond UNIX – A True Distributed System for the 1990s

Andrew S. Tanenbaum
Robbert van Renesse
Hans van Staveren
Gregory J. Sharp
Dept. of Mathematics and Computer Science
Vrije Universiteit
De Boelelaan 1081
1081 HV Amsterdam, The Netherlands
ast@cs.vu.nl, cogito@cs.vu.nl, sater@cs.vu.nl, gregor@cs.vu.nl

UNIX has been around now for almost 20 years. At the time UNIX began, most departments felt themselves well-endowed indeed if they owned a single PDP-11/45 with 256K memory and a 2.5M RK05 disk. Nowadays a laptop would be embarrassed to have only that. It is our hypothesis that UNIX is no longer the appropriate kind of operating system for the 1990s. In this paper, a new system, Amoeba, will be described, that we believe meets the requirements for distributed computing in the 1990s.

rcc – an optimising C compiler for the Motorola 88000

Ciaran O'Donnell
Bureau d'Etudes O'Donnell
Centre National d'Etudes de Telecommunications
France

Historically UNIX C compilers have been influenced by three design goals: – a "robust" "straightforward" internal organisation to permit compiler "maintenance by a small research group. – reasonably good code quality with the programmer bearing some of the responsibility for optimisation. – a "portable" organisation.

The introduction of RISC in machine architecture, while simplifying some aspects of compiler design, complicates the rest. Although template selection is simplified by the limited number of instructions and addressing modes, the compiler builder must now address issues such as: – Automatic register allocation for automatics – Use/def chains for local and global common sub-expressions – The effect of cache on execution time, and – Instruction re-ordering. – This evolution poses the question: is it still possible to address the efficiency goal for these new architectures, while maintaining a straightforward, robust, and portable internal organisation?

In this paper we describe the implementation of the C compiler "rcc" for a new fault tolerant computer architecture being developed by France Telecom's research organisation (CNET). The compiler generates excellent code for a particularly sophisticated RISC processor: the Motorola 88000. The compiler was developed from scratch in less than a year and accepts more or less complete ANSI C. The source code is currently less than 14000 lines long.

We describe the asymmetric "call/return" calling sequence used by the compiler. This scheme allows good execution time, and excellent code density, even for non-leaf procedures.

Some of the issues involved in porting of the compiler to another RISC architecture, are discussed.

***Limits* – A System for UNIX Resource Administration**[†]

Andrew Bettison

Peter Chubb
Andrew Gollan
Chris Maltby
Greg Rose
Neil Russell

Softway Pty Ltd
79, Myrtle Street
Chippendale
NSW 2008
Australia
greg@softway.oz.au

The UNIX operating system, despite its emergence as a standard for supercomputer systems, lacks effective support for multiuser resource administration. The design and implementation of a decentralised resource administration system uniformly realisable across the wide variety of UNIX dialects presents a number of problems. Among these problems are potential violations of the UNIX design philosophy, preservation of the user process environment and adherence to industry standards.

A research project to design and develop a UNIX resource administration system built upon the "Fair Share" scheduler, *Share*, has been undertaken at Softway with the support of the Australian Government. This project has yielded a resource administration and control system called *Limits* and further research is being undertaken on its extension to networks supporting shared resources.

A Pageable Memory Based Filesystem[†]

Marshall Kirk McKusick
Michael J. Karels
Keith Bostic

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720
mckusick@okeeffe.Berkeley.EDU

This paper describes the motivations for memory-based filesystems. It compares techniques used to implement them and describes the drawbacks of using dedicated memory to support such filesystems. To avoid the drawbacks of using dedicated memory, it discusses building a simple memory-based filesystem in pageable memory. It details the performance characteristics of this filesystem and concludes with areas for future work.

A New Framework for Device Support in Berkeley UNIX

Chris Torek

Computer Science Department,
University of Maryland
College Park,
MD 20742
chris@mimsy.umd.edu

Device independence is a fundamental aspect of the UNIX operating system. UNIX systems represent devices using the same "stream of bytes" model they give to files. Hiding the particulars of devices can be a complex task, and some of the work is relegated to the device driver routines for each device, in the machine dependent portion of the kernel; but the system provides a body of machine independent common code as well. This improves uniformity and reduces code duplication. For instance, existing Berkeley UNIX systems handle serial ports by grouping them together as "tty devices", and provide "line discipline" routines to deal with terminal state; they hide disk sectoring behind a "block I/O system" that deals specifically with disk drives (and, to a limited extent, tape drives as well). The machine independent routines create, in effect, "device classes" within the UNIX kernel. For system configuration, however, the machine dependent part of the kernel defines a different set of classes based on the location of each device on the machine's I/O busses. Although they should be orthogonal, these two classes are in conflict. Currently the machine dependent classes come first: the system configuration and analysis programs know about the hardware. This paper describes a plan to formalise and extend the device classes, and to remove the hardware knowledge from the system configuration and monitoring programs.

Pictures of Programs

Jon Bentley

jlb@research.att.com

A good picture can indeed be worth a kiloword. While computers have revolutionized images in fields ranging from radiology to meteorology to abstract topology, most programmers make little use of computer-generated pictures. This paper shows how UNIX tools for drawing pictures can help us to understand the behavior of some computer programs.

Symphony Emulation and the Ultravirtuoso

Michael Hawley

MIT Media Laboratory,
20 Ames Street,
Cambridge MA 02139
mike@tome.media.mit.edu

"Typical" musical control information – the instrumental gesture – flows at about 3Kbytes per minute per soloist, or 10-20Mb per composer per career. In a sense this represents compression by about 10000:1 from CD-quality audio. All Rachmaninoff's piano music occupies about 2Mb, and a CD-Rom of music coded this way lasts more than a year of continuous playback. Instrument synthesis is also evolving (current synthesizers are like the clavichords of the late 20th century), and recently we enhanced our digital Bosendorfer grand piano to communicate in MIDI, the synthesizer protocol. This makes for a fun ensemble: a synthetic orchestra and a super piano.

This paper discusses recent music work. Among other things, we tried a "bed of nails" orchestration test: the complete synthesis of a high-romantic piano concerto, Franz Liszt's *Totentanz*. The idea was to produce a thrilling performance in the grand orchestral tradition – without any real players. I will discuss in detail the *Totentanz* and the technology used to make it, with some reflections on the process.

"Foiling the Cracker": A Survey of, and Improvements to, Password Security[†]

Daniel V. Klein

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15217
dvk@sei.cmu.edu

With the rapid burgeoning of national and international networks, the question of system security has become one of growing importance. High speed inter-machine communication and even higher speed computational processors have made the threats of system "crackers", data theft, data corruption very real. This paper outlines some of the problems of current password security by demonstrating the ease by which individual accounts may be broken. Various techniques used by crackers are outlined, and finally one solution to this point of system vulnerability, a proactive password checker, is proposed.

UNIX and Supercomputers

Greg Rose

Softway Pty Ltd
79, Myrtle Street
Chippendale
NSW 2008
Australia

Since its quiet beginnings around 1970, UNIX has pervaded a very wide range of computer hardware, from small micros to very large mainframes, and recently, supercomputers. This paper attempts to examine the place of UNIX in the supercomputer world, along with the features of UNIX which are relevant to that niche, both good and bad.

The fact that an interesting and fast developing segment of the marketplace is rapidly adopting UNIX is reflecting itself in changes to the future direction of UNIX itself. Current efforts to better adapt UNIX to supercomputers, and relevant standards, are described.

UNIX has brought more cooperation and networking into the world of supercomputers, and is starting to change the way we are viewing that world. This paper will also prophesy future directions of supercomputers as influenced by UNIX.

More Taste: Less Greed?
Sending UNIX to the Fat Farm

C H Forsyth

Department of Computer Science
University of York
Heslington
York YO1 5DD
England
forsyth@minster.york.ac.uk

You (like us) have 80 to 90 Sun 3/50 machines with 4 megabytes of memory. You have been given some optical discs containing System V.4. Which can you least afford to discard? Things are getting out of hand. Soon, 32 megabyte Ultrasparks will be needed just for us to start the window system in under a minute.

For UNIX, now in middle-age rotundly recalling its sprightly youth, mere "tuning" will not cause that heavy code to slip away. We need to reconsider and re-implement the system interface periodically, to take account of changes in its environment. We must be willing to *throw things away*, discarding parts of the older implementation completely, rather than corrupting clean new mechanisms to approximate the mistakes of the past.

To illustrate this thesis, I shall discuss work I have done on SunOS 3.5 to reduce its size and complexity. The virtual memory system has been replaced by a simpler one using ideas from the EMAS system and elsewhere. A stream I/O system in the 8th/9th Edition style has been added, replacing the old character I/O system, pipes, and UNIX domain sockets. I have also made some preliminary forays into the Virtual File System interface.

The Case Against UNIX Standards

Robert Swartz

Mark Williams Company
601 North Skokie Highway
Lake Buff
Illinois 60044
USA
rs@mwc.uu.net

Recent releases of UNIX are too large and complicated to be useful. This is due to the standardization process. This process has attempted to produce one version of the system for all applications, which is not feasible. The committee method of producing software specifications produces systems which are seriously flawed. Standardization acts to retard innovation. Instead of attempting to produce a monolithic UNIX standard we should have a number of competing versions and let the market decide which systems they wish to choose.

Standards, Specifications and Open Systems

Peter H. Salus

The Sun User Group, Inc.
peter@uunet.uu.net

The existence of standards and industrial specifications which will be recognized by a wide range of hardware and software suppliers is vital to the success of "open systems." Though feared by the research and advanced development community, national and international standards are the level ground upon which future work can be erected, not inhibitory prescriptions nor deterrents to future directions.

Plan 9 from Bell Labs

Rob Pike
Dave Presotto
Ken Thompson
Howard Trickey

rob@research.att.com

Plan 9 is a distributed computing environment. It is assembled from separate machines acting as CPU servers, file servers, and terminals. The pieces are connected by a single file-oriented protocol and local name space operations. By building the system from distinct, specialised components rather than from similar general-purpose components, Plan 9 achieves levels of efficiency, security, simplicity, and reliability seldom realised in other distributed systems. This paper discusses the building blocks, interconnections, and conventions of Plan 9.

Multiprocessor Streams for Plan 9

David Leo Presotto

presotto@research.att.com

This paper describes an implementation of Streams for the Plan 9 kernel, a multi-threaded, multiprocessor kernel with a system call interface reminiscent of UNIX. Rather than port Dennis Ritchie's Streams to Plan 9, we changed the abstraction to fit more naturally into the new environment. The result is a

mechanism that has similar performance and is internally easier to program.

Rc – A Shell for Plan 9 and UNIX Systems

Tom Duff

td@research.att.com

Rc is a command interpreter for Plan 9. It also runs on a variety of traditional systems, including SunOS and the Tenth Edition. It provides similar facilities to Bourne's */bin/sh*, with some small additions and mostly less idiosyncratic syntax. This paper introduces *rc*'s highlights with numerous examples, and discusses its design and why it varies from Bourne's.

Variable-Size Arrays in C

Dennis M. Ritchie

dmr@research.att.com

The type system of the C language admits arrays of constant size only. The language's rules for pointers make this limitation bearable for vectors, but leave it difficult to write routines that handle multi-dimensional arrays whose bounds may vary. This paper discusses the difficulties arising from the most obvious generalization, which is to permit arrays whose bounds are not constant expressions. It proposes an alternate extension: array pointers that carry the bounds within themselves. The proposal is less injurious to the existing type system of the language than others, and retains the property that a pointer's value, type, and offset suffice to determine how to access the value referred to.

A New C Compiler

Ken Thompson

ken@research.att.com

This paper describes yet another series of C compilers. These compilers were developed over the last several years and are now in use on Plan 9. These compilers are experimental in nature and were developed to try out new ideas. Some of the ideas were good and some not so good.

Does C++ Really Need Multiple Inheritance?

T. A. Cargill

P.O. Box 69, Louisville,
Colorado 80027,
USA
cargill@spot.colorado.edu

Multiple inheritance was introduced to C++ [Str86a] in version 2.0 of the Cfront implementation [AT&89a]. The potential benefits of multiple inheritance (MI) must be weighed against the increased language complexity and implementation overhead incurred. In search of these benefits I have examined several published examples of MI in C++ [Wie89a, Lip89a, Dew89a, Sho89a]. In all but one case, the use of MI is not essential to the constructed program; single inheritance (SI) is adequate, and (subjectively) yields simpler code. The easiest way to eliminate MI is to replace inheritance by embedding an object rather than inheriting from its class. For the example that does not succumb to this technique the SI version involves the duplication of a little code. This one candidate for legitimate use of MI is a complex piece of code, the product of long design exercise by some of the most experienced C++ programmers; it is optimistic to expect such efforts to be applied to most programs. If the legitimate use of MI remains the exception, the conclusion must be that its inclusion in C++ is of doubtful utility.

group: A Distributed Group Specification and Management Service

Thomas J. Bannon
Ivor P. Page

The University of Texas at Dallas
Dallas
bannon@csc.ti.com

This paper presents the design of a distributed service to support collaborative interaction among groups of computer users spread out across a network. The model of group membership, the service architecture, and the design for system security are described in some detail. The design and implementation of the service are biased toward computers running the UNIX operating system and the TCP/IP protocol, but they should not prove too difficult to port to environments of similar capability.

N(e)FS: the Protocol is the Problem

Jim Reid

Department of Computer Science,
Strathclyde University,
Richmond Street,
Glasgow,
Scotland,
G1 1XH.
jim@cs.strath.ac.uk

The Network File System protocol (NFS) developed by Sun Microsystems has emerged as the *de facto* standard for sharing filesystems amongst UNIX systems. Indeed, it has also become popular on other operating systems such as DOS and VMS.

A cornerstone of the NFS protocol is the notion of a stateless server. File servers simply process each NFS request they receive in isolation from previous requests sent by clients. This statelessness considerably simplifies server operations since NFS servers need not keep track of what individual clients are doing. Stateless servers also obviate the need for special crash recovery techniques by the server.

Despite these obvious attractions, the realisation of stateless NFS servers also presents several problems. Sun have announced a new protocol – NeFS (Network Extensible File System) – in an attempt to solve some of these problems. In this paper, weaknesses in both NFS and NeFS are discussed and some possible solutions outlined.

AUSTRIA - UUGA
Friedrich Kofler
Schottenring 33/Hof
A-1010 Wien
AUSTRIA
Tel: +43 222 34 61 84
uuga@tuvie.at

BELGIUM - BUUG
Marc Nyssen
Department of Medical Informatics
VUB, Laarbeeklaan 103
B-1090 Brussels
BELGIUM
Tel: +32 2 477 4111
Fax: +32 2 477 4000
buug@vub.uucp

DENMARK - DKUUG
Mogens Buhelt
Kabbelejevej 27B
DK-2700 Bronshoj
DENMARK
Tel: +45 31 60 65 80
mogens@dkuug.dk

FINLAND - FUUG
Anu Patrikka-Cantell
Finnish UNIX Users' Group
Arkadiankatu 14 B 45
00100 Helsinki
FINLAND
Tel: +358 0 494 371

FRANCE - AFUU
Miss Ann Garnery
AFUU
11 rue Carnot
94270 Le Kremlin Bicetre
Paris
FRANCE
Tel: +33 1 46 70 95 90
Fax: +33 1 46 58 94 20
anne@afuu.fr

GERMANY - GUUG
Dr Anton Gerold
GUUG-Vorstand
Elsenheimerstr 43
D-8000 MUNCHEN 21
WEST GERMANY
Tel: +49 89 570 7697
Fax: +49 89 570 7607
gerold@lan.informatik.tu-muenchem.dbp.de

HUNGARY - HUUG
Dr Elôd Knuth
Computer and Automation Institute
Hungarian Academy of Sciences
H-1502 Budapest 112, P.O. Box 63
HUNGARY
Tel: +361 1 665 435
Fax: +361 1 354 317

ICELAND - ICEUUG
Marius Olafsson
University Computer Center
Dunhaga 5
Reykjavik
ICELAND
Tel: +354 1 694747
marius@rhi.hi.is

IRELAND - IUUG
Norman Hull
Irish UNIX-systems User Group
Court Place
Carlow
IRELAND
Tel: +353 503 31745
Fax: +353 503 43695
iuug-exec@cs.tcd.ie

ITALY - i2u
Ing Carlo Mortarino
i2u
Viale Monza 347
20126 Milano
ITALY
Tel: +39 2 2520 2530

NETHERLANDS - NLUUG
Patricia Otter
p/a Xirion bv
Burg. Verderlaan 15 X
3454 PE De Meern
THE NETHERLANDS
Tel: +31 3406 61 990
nluug@hp4nl

NORWAY - NUUG
Jan Brandt Jensen
Unisoft A.S.
Enebakkvn 154
N-0680 Oslo 6
NORWAY
Tel: +47 2 688970

PORTUGAL - PUUG
Legatheaux Martens
Avenue 24 de Julho, nº 134, 7º
Lisboa
PORTUGAL
Tel: +351 1 673194/609822
Fax: +351 1 7597716
puug@inesc.pt

EUUG-S
N
I
X

SWEDEN - EUUG-S
Hans E. Johansson
NCR Svenska AB
Box 1206
S-164 28 KISTA
SWEDEN
Tel: +46 8 750 26 03
hans@ncr.se

UNITED KINGDOM - UKUUG
Bill Barrett
Owles Hall
Buntingford
Hertfordshire SG9 9PL
UNITED KINGDOM
Tel: +44 763 73039
Fax: +44 763 73255
ukuug@ukc.ac.uk

The European UNIX systems User Group
Owles Hall
Buntingford
Hertfordshire SG9 9PL
UNITED KINGDOM
Tel: +44 763 73039
Fax: +44 763 73255
euug@EU.net

CZECHOSLOVALIA - CSUUG
Sekretariat CSUUG
Výpočetni Centrum Vše
Nam. A. Zápotockého4
130 67 PRAKA 3
CZECHOSLOVALIA
csuug@vse.uucp

YUGOSLAVIA - YUUG
Milan Palian
Iskra Delta Computers
Parmova 41
61000 Ljubljana
YUGOSLAVIA
Tel: +38 61 574 554
mpalian@idcyuug.uucp