Current Research by
The Computer Systems Research Group
of Berkeley

Marshall Kirk McKusick
Michael J Karels
Keith Sklower
Kevin Fall
Marc Teitelbaum
Keith Bostic

Summary

The release of 4.3BSD in April of 1986 addressed many of the performance problems and unfinished interfaces present in 4.2BSD [Leffler84] [McKusick85]. The Computer Systems Research Group at Berkeley has now embarked on a new development phase to update other major components of the system, as well as to offer new functionality. There are five major ongoing projects. The first is to develop an OSI network protocol suite and to integrate existing ISO applications into Berkeley UNIX. The second is to develop and support an interface compliant with the P1003.1 POSIX standard recently approved by the IEEE. The third is to refine the TCP/IP networking to improve its performance and limit congestion on slow and/or lossy networks. The fourth is to provide a standard interface to file systems so that multiple local and remote file systems can be supported, much as multiple networking protocols are supported by 4.3BSD. The fifth is to evaluate alternate access control mechanisms and audit the existing security features of the system, particularly with respect to network services. Other areas of work include multi-architecture support, a general purpose kernel memory allocator, disk labels, and extensions to the 4.2BSD fast filesystem.

We are planning to finish implementation prototypes for each of the five main areas of work over the next year, and provide an informal test release sometime next year for interested developers. After incorporating feedback and refinements from the testers, they will appear in the next full Berkeley release, which is typically made about a year after the test release.

1. Recently Completed Projects

There have been several changes in the system that were included in the recent 4.3BSD Tahoe release.

1.1. Multi-architecture support

Support has been added for the DEC VAX 8600/8650, VAX 8200/8250, MicroVAXII and MicroVAXIII.

The largest change has been the incorporation of support for the first non-VAX processor, the CCI Power 6/32 and 6/32SX. (This addition also supports the Harris HCX-7 and HCX-9, as well as the Sperry 7000/40 and ICL machines.) The Power 6 version of 4.3BSD is largely based on the compilers and device drivers done for CCI's 4.2BSD UNIX, and is otherwise similar to the VAX release of 4.3BSD. The entire source tree, including all kernel and user-level sources, has been merged using a structure that will easily accommodate the addition of other processor families. A MIPS R2000 has been donated to us, making the MIPS architecture a likely candidate for inclusion into a future BSD release.

1.2. Kernel Memory Allocator

The 4.3BSD UNIX kernel used 10 different memory allocation mechanisms, each designed for the particular needs of the utilizing subsystem. These mechanisms have been replaced by a general purpose dynamic memory allocator that can be used by all of the kernel subsystems. The design of this allocator takes advantage of known memory usage patterns in the UNIX kernel and a hybrid strategy that is time-efficient for small allocations and space-efficient for large allocations. This allocator replaces the multiple memory allocation interfaces with a single easy-to-program interface, results in more efficient use of global memory by eliminating partitioned and specialized memory pools, and is quick enough (approximately 15 VAX instructions) that no performance loss is observed relative to the current implementations. [McKusick88].

1.3. Disk Labels

During the work on the CCI machine, it became obvious that disk geometry and filesystem layout information must be stored on each disk in a pack label. Disk labels were implemented for the CCI disks and for the most common types of disk controllers on the VAX. A utility was written to create and maintain the disk information, and other user-level programs that use such information now obtain it from the disk label. The use of this facility has allowed improvements in the file system's knowledge of irregular disk geometries such as track-to-track skew.

1.4. Fat Fast File System

The 4.2 fast file system [McKusick84] contained several statically sized structures, imposing limits on the number of cylinders per cylinder group, inodes per cylinder group, and number of distinguished rotational positions. The new "fat" filesystem allows these limits to be set at filesystem creation time. Old kernels will treat the new filesystems as read-only, and new kernels will accommodate both formats. The filesystem check facility, `fsck`, has also been modified to check either type.

2. Current UNIX Research at Berkeley

Since the release of 4.3BSD in mid 1986, we have begun work on several new major areas of research. Our goal is to apply leading edge research ideas into a stable and reliable implementation that solves current problems in operating systems development.

2.1. OSI network protocol development

The network architecture of 4.2BSD was designed to accommodate multiple network protocol families and address formats, and an implementation of the ISO OSI network protocols should enter into this framework without much difficulty. We plan to implement the OSI connectionless internet protocol (CLNP), and device drivers for X.25, 802.3, and possibly 802.5 interfaces, and to integrate these with an OSI transport class 4 (TP-4) implementation. We will also incorporate into the Berkeley Software Distribution an updated ISO Development Environment (ISODE) featuring International Standard (IS) versions of utilities. ISODE implements the session and presentation layers of the OSI protocol suite, and will include an implementation of the file transfer protocol (FTAM). It is also possible that an X.400 implementation now being done at University College, London and the University of Nottingham will be available for testing and distribution.

This implementation is comprised of four areas.

1) We are updating the University of Wisconsin TP-4 to match GOSIP requirements. The University of Wisconsin developed a transport class 4 implementation for the 4.2BSD kernel under contract to Mitre. This implementation must be updated to reflect the National Institute of Standards and Technology (NIST, formerly NBS) workshop agreements, GOSIP, and 4.3BSD requirements. We will make this TP-4 operate with an OSI IP, as the original implementation was built to run over the DoD IP.

2) A kernel version of the OSI IP and ES-IS protocols must be produced. We will implement the kernel version of these protocols.

3) The required device drivers need to be integrated into a BSD kernel. 4.3BSD has existing device drivers for many Ethernet devices; future BSD versions may also support X.25 devices as well as token ring networks. These device drivers must be integrated into the kernel OSI protocol

implementations.

4) The existing OSINET interoperability test network is available so that the interoperability of the ISODE and BSD kernel protocols can be established through tests with several vendors. Testing is crucial because an openly available version of GOSIP protocols that does not interoperate with DEC, IBM, SUN, ICL, HIS, and other major vendors would be embarrassing. To allow testing of the integrated pieces the most desirable approach is to provide access to OSINET at UCB. A second approach is to do the interoperability testing at the site of an existing OSINET member, such as the NBS.

2.2. Compliance with POSIX 1003

Berkeley became involved several months ago in the development of the IEEE POSIX P1003.1 system interface standard. Since then, we have been participating in the working groups of P1003.2 (shell and application utility interface), P1003.6 (security), P1003.7 (system administration), and P1003.8 (networking).

The IEEE published the POSIX P1003.1 standard in late 1988. POSIX related changes to the BSD system have included a new terminal driver, support for POSIX sessions and job control, expanded signal functionality, restructured directory access routines, and new set-user and set-group id facilities. We currently have a prototype implementation of the POSIX driver with extensions to provide binary compatibility with applications developed for the old Berkeley terminal driver. We also have a prototype implementation of the 4.2BSD-based POSIX job control facility.

The P1003.2 draft is currently being voted on by the IEEE P1003.2 balloting group. Berkeley is particularly interested in the results of this standard, as it will profoundly influence the user environment. The other groups are in comparatively early phases, with drafts coming to ballot sometime in the 90's. Berkeley will continue to participate in these groups, and move in the near future toward a P1003.1 and P1003.2 compliant system. We have many of the utilities outlined in the current P1003.2 draft already implemented, and have other parties willing to contribute additional implementations.

2.3. Improvements to the TCP/IP Networking Protocols

The Internet and the Berkeley collection of local-area networks have both grown at high rates in the last year. The Bay Area Regional Research Network (BARRNet), connecting several UC campuses, Stanford and NASA-Ames has recently become operational, increasing the complexity of the network connectivity. Both Internet and local routing algorithms are showing the strain of continued growth. We have made several changes in the local routing algorithm to keep accommodating the current topology, and are participating in the development of new routing algorithms and standard protocols.

Recent work in collaboration with Van Jacobson of the Lawrence Berkeley Laboratory has led to the design and implementation of several new algorithms for TCP that improve throughput on both local and long-haul networks while reducing unnecessary retransmission. The improvement is especially striking when connections must traverse slow and/or lossy networks. The new algorithms include "slow-start," a technique for opening the TCP flow control window slowly and using the returning stream of acknowledgements as a clock to drive the connection at the highest speed tolerated by the intervening network. A modification of this technique allows the sender to dynamically modify the send window size to adjust to changing network conditions. In addition, the round-trip timer has been modified to estimate the variance in round-trip time, thus allowing earlier retransmission of lost packets with less spurious retransmission due to increasing network delay. Along with a scheme proposed by Phil Karn of Bellcore, these changes reduce unnecessary retransmission over difficult paths such as Satnet by nearly two orders of magnitude while improving throughput dramatically.

The current TCP implementation is now being readied for more widespread distribution via the network and as a standard Berkeley distribution unencumbered by any commercial licensing. We are continuing to refine the TCP and IP implementations using the ARPANET, BARRNet, the NSF network and local campus nets as testbeds. In addition, we are incorporating applicable algorithms from this work into the TP-4 protocol implementation.

2.4. Toward a Compatible File System Interface

The most critical shortcoming of the 4.3BSD UNIX system was in the area of distributed file systems. As with networking protocols, there is no single distributed file system that provides sufficient speed and functionality for all problems. It is frequently necessary to support several different remote file system protocols, just as it is necessary to run several different network protocols.

As network or remote file systems have been implemented for UNIX, several stylized interfaces between the file system implementation and the rest of the kernel have been developed. Among these are Sun Microsystems' Virtual File System interface (VFS) using **vnodes** [Sandburg85] [Kleiman86], Digital Equipment's Generic File System (GFS) architecture [Rodriguez86], AT&T's File System Switch (FSS) [Rifkin86], the LOCUS distributed file system [Walker85], and Masscomp's extended file system [Cole85]. Other remote file systems have been implemented in research or university groups for internal use, notably the network file system in the Eighth Edition UNIX system [Weinberger84] and two different file systems used at Carnegie Mellon University [Satyanarayanan85]. Numerous other remote file access methods have been devised for use within individual UNIX processes, many of them by modifications to the C I/O library similar to those in the Newcastle Connection [Brownbridge82].

Each design attempts to isolate file system-dependent details below a generic interface and to provide a framework within which new file systems may be incorporated. However, each of these interfaces is different from and incompatible with the others. Each addresses somewhat different design goals, having been based on a different version of UNIX, having targeted a different set of file systems with varying characteristics, and having selected a different set of file system primitive operations.

Our effort in this area is aimed at providing a common framework to support these different distributed file systems simultaneously rather than to simply implement yet another protocol. This requires a detailed study of the existing protocols, and discussion with their implementors to determine whether they could modify their implementation to fit within our proposed framework. We have studied the various file system interfaces to determine their generality, completeness, robustness, efficiency, and aesthetics and are currently working on a file system interface that we believe includes the best features of each of the existing implementations. This work and the rationale underlying its development have been presented to major software vendors as an early step toward convergence on a standard compatible file system interface. Briefly, the proposal adopts the 4.3BSD calling convention for file name lookup but otherwise is closely related to Sun's VFS and DEC's GFS. [Karels86].

2.5. System Security

The recent invasion of the DARPA Internet by a quickly reproducing ‘‘worm’’ highlighted the need for a thorough review of the access safeguards built into the system. Until now, we have taken a passive approach to dealing with weaknesses in the system access mechanisms, rather than actively searching for possible weaknesses. When we are notified of a problem or loophole in a system utility by one of our users, we have a well defined procedure for fixing the problem and expeditiously disseminating the fix to the BSD mailing list. This procedure has proven itself to be effective in solving known problems as they arise (witness its success in handling the recent worm). However, we feel that it would be useful to take a more active role in identifying problems before they are reported (or exploited). We will make a complete audit of the system utilities and network servers to find unintended system access mechanisms.

As a part of the work to make the system more resistant to attack from local users or via the network, it will be necessary to produce additional documentation on the configuration and operation of the system. This documentation will cover such topics as file and directory ownership and access, network and server configuration, and control of privileged operations such as file system backups.

We are investigating the addition of access control lists (ACLs) for filesystem objects. ACLs provide a much finer granularity of control over file access permissions than the current discretionary access control mechanism (mode bits). Furthermore, they are necessary in environments where C2 level security or better, as defined in the DoD TCSEC [DoD83], is required. The POSIX P1003.6 security group has made notable progress in determining how an ACL mechanism should work, and several vendors have implemented ACLs for their commercial systems. Berkeley will investigate the existing implementations and determine how to best integrate ACLs with the existing mechanism.

A major shortcoming of the present system is that authentication over the network is based solely on the privileged port mechanism between trusting hosts and users. Although privileged ports can only be created by processes running as root on a UNIX system, such processes are easy for a workstation user to obtain; they simply reboot their workstation in single user mode. Thus, a better authentication mechanism is needed. At present, we believe that the MIT Kerberos authentication server [Steiner88] provides the best solution to this problem. We propose to investigate Kerberos further as well as other authentication mechanisms and then to integrate the best one into Berkeley UNIX. Part of this integration would be the addition of the authentication mechanism into utilities such as telnet, login, remote shell, etc. We will add support for telnet (eventually replacing rlogin), the X window system, and the mail system within an authentication domain (a Kerberos *realm*). We hope to replace the existing password authentication on each host with the network authentication system.

3. References

Brownbridge82

Brownbridge, D.R., L.F. Marshall, B. Randell, ‘‘The Newcastle Connection, or UNIXes of the World Unite!,’’ *Software− Practice and Experience*, Vol. 12, pp. 1147-1162, 1982.

Cole85

Cole, C.T., P.B. Flinn, A.B. Atlas, ‘‘An Implementation of an Extended File System for UNIX,’’ *Usenix Conference Proceedings*, pp. 131-150, June, 1985.

DoD83

Department of Defense, ‘‘Trusted Computer System Evaluation Criteria,’’ *CSC-STD-001-83*, DoD Computer Security Center, August, 1983.

Karels86

Karels, M., M. McKusick, ‘‘Towards a Compatible File System Interface,’’ *Proceedings of the European UNIX Users Group Meeting*, Manchester, England, pp. 481-496, September 1986.

Kleiman86

Kleiman, S., ‘‘Vnodes: An Architecture for Multiple File System Types in Sun UNIX,’’ *Usenix Conference Proceedings*, pp. 238-247, June, 1986.

Leffler84

Leffler, S., M.K. McKusick, M. Karels, ‘‘Measuring and Improving the Performance of 4.2BSD,’’ *Usenix Conference Proceedings*, pp. 237-252, June, 1984.

McKusick84

McKusick, M.K., W. Joy, S. Leffler, R. Fabry, ‘‘A Fast File System for UNIX’’, *ACM Transactions on Computer Systems 2*, 3. pp 181-197, August 1984.

McKusick85

McKusick, M.K., M. Karels, S. Leffler, ‘‘Performance Improvements and Functional Enhancements in 4.3BSD,’’ *Usenix Conference Proceedings*, pp. 519-531, June, 1985.

McKusick86

McKusick, M.K., M. Karels, ‘‘A New Virtual Memory Implementation for Berkeley UNIX,’’ *Proceedings of the European UNIX Users Group Meeting*, Manchester, England, pp. 451-460, September 1986.

McKusick88
McKusick, M.K., M. Karels, ‘‘Design of a General Purpose Memory Allocator for the 4.3BSD UNIX Kernel,’’ *Usenix Conference Proceedings*, pp. 295-303, June, 1988.

Rifkin86
Rifkin, A.P., M.P. Forbes, R.L. Hamilton, M. Sabrio, S. Shah, K. Yueh, ‘‘RFS Architectural Overview,’’ *Usenix Conference Proceedings*, pp. 248-259, June, 1986.

Rodriguez86
Rodriguez, R., M. Koehler, R. Hyde, ‘‘The Generic File System,’’ *Usenix Conference Proceedings*, pp. 260-269, June, 1986.

Sandberg85
Sandberg, R., D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, ‘‘Design and Implementation of the Sun Network File System,’’ *Usenix Conference Proceedings*, pp. 119-130, June, 1985.

Satyanarayanan85
Satyanarayanan, M., *et al.*, ‘‘The ITC Distributed File System: Principles and Design,’’ *Proc. 10th Symposium on Operating Systems Principles*, pp. 35-50, ACM, December, 1985.

Steiner88
Steiner, J., C. Newman, J. Schiller, ‘‘*Kerberos:* An Authentication Service for Open Network Systems,’’ *Usenix Conference Proceedings*, pp. 191-202, February, 1988.

Walker85
Walker, B.J. and S.H. Kiser, ‘‘The LOCUS Distributed File System,’’ *The LOCUS Distributed System Architecture*, G.J. Popek and B.J. Walker, ed., The MIT Press, Cambridge, MA, 1985.

Weinberger84
Weinberger, P.J., ‘‘The Version 8 Network File System,’’ *Usenix Conference presentation*, June, 1984.