

MARCUS BENNETT

JULY-AUGUST 1978
VOL. 57, NO. 6, PART 2

THE BELL SYSTEM TECHNICAL JOURNAL



ISSN0005-8580

UNIX TIME-SHARING SYSTEM

T. H. Crowley	Preface	1897
M. D. McIlroy, E. N. Pison, and B. A. Tague	Foreword	1899
D. M. Ritchie and K. Thompson	The UNIX Time-Sharing System	1905
K. Thompson	UNIX Implementation	1931
D. M. Ritchie	A Retrospective	1947
S. R. Bourne	The UNIX Shell	1971
D. M. Ritchie, S. C. Johnson, M. E. Lesk, and B. W. Kernighan	The C Programming Language	1991
S. C. Johnson and D. M. Ritchie	Portability of C Programs and the UNIX System	2021
H. Lycklama and D. L. Bayer	The MERT Operating System	2049
H. Lycklama	UNIX on a Microprocessor	2087
H. Lycklama and C. Christensen	A Minicomputer Satellite Processor System	2103
B. W. Kernighan, M. E. Lesk, and J. F. Ossanna, Jr.	Document Preparation	2115
L. E. McMahon, L. L. Cherry, and R. Morris	Statistical Text Processing	2137
S. C. Johnson and M. E. Lesk	Language Development Tools	2155
T. A. Dolotta, R. C. Haight, and J. R. Mashey	The Programmer's Workbench	2177

(Contents continued on outside back cover)

THE BELL SYSTEM TECHNICAL JOURNAL

ADVISORY BOARD

D. E. PROCKNOW, *President, Western Electric Company, Incorporated*

W. O. BAKER, *President, Bell Telephone Laboratories, Incorporated*

C. L. BROWN, *President, American Telephone and Telegraph Company*

EDITORIAL COMMITTEE

D. GILLETTE, *Chairman*

W. S. BOYLE

I. DORROS

A. G. CHYNOWETH

H. B. HEILIG

S. J. BARBERA

C. B. SHARP

T. H. CROWLEY

B. E. STRASSER

W. A. DEPP

I. WELBER

EDITORIAL STAFF

G. E. SCHINDLER, JR., *Editor*

J. B. FRY, *Associate Editor*

H. M. PURVIANCE, *Art and Production*

H. K. LINDEMANN, *Circulation*

S. P. MORGAN, *Coordinating Editor of UNIX Time-Sharing System*

THE BELL SYSTEM TECHNICAL JOURNAL is published monthly, except for the May-June and July-August combined issues, by the American Telephone and Telegraph Company, J. D. deButts, Chairman and Chief Executive Officer; C. L. Brown, President; W. G. Burns, Vice President and Treasurer; F. A. Hutson, Jr., Secretary. Editorial enquiries should be addressed to the Editor, The Bell System Technical Journal, Bell Laboratories, 600 Mountain Ave., Murray Hill, N.J. 07974. Checks for subscriptions should be made payable to The Bell System Technical Journal and should be addressed to Bell Laboratories, Circulation Group, Whippany Road, Whippany, N.J. 07981. Subscriptions \$20.00 per year; single copies \$2.00 each. Foreign postage \$1.00 per year; 15 cents per copy. Printed in U.S.A. Second-class postage paid at New Providence, New Jersey 07974 and additional mailing offices.

© 1978 American Telephone and Telegraph Company

Single copies of material from this issue of the Bell System Technical Journal may be reproduced for personal, noncommercial use. Permission to make multiple copies must be obtained from the editor.

Mine of Information Ltd.,
1 Francis Avenue,

IPR2025-01400
CrowdStrike EX1036 Page 2

THE BELL SYSTEM TECHNICAL JOURNAL

DEVOTED TO THE SCIENTIFIC AND ENGINEERING
ASPECTS OF ELECTRICAL COMMUNICATION

Volume 57

July–August 1978

Number 6, Part 2

Copyright © 1978 American Telephone and Telegraph Company. Printed in U.S.A.

UNIX Time-Sharing System:

Preface

By T. H. CROWLEY

(Manuscript received April 18, 1978)

Since 1962, *The Bell System Technical Journal* has published over 90 articles on computer programming. Although that number is not insignificant, it is only about 6 percent of all the articles published in the B.S.T.J. during that period. Publications in the B.S.T.J. tend to reflect the amount of activity in many areas of technology at Bell Laboratories, but that has certainly not been true for computer programming work. Better indicators of the importance of programming for current Bell Laboratories work are the following:

- (i) 25 percent of the technical staff spent more than 50 percent of their time on programming, or related work, in 1977.
- (ii) 25 percent of the professional staff recruited in 1977 majored in computer science.
- (iii) 40 percent of the employees entering the Bell Laboratories Graduate Study Program in 1977 are majoring in computer science.

Programming activities under way at Bell Laboratories cover a very broad spectrum. They range from basic research on compiler-

generating techniques to the maintenance of Bell Laboratories-developed programs now in routine use at operating telephone companies. They include writing of real-time control programs for switching systems, development of time-shared text editing facilities, and design of massive data-base systems. They involve work on microprocessors, minicomputers, and maxicomputers. They extend from the design of sophisticated programming tools to be used only by experts to the delivery of program products to be used by clerks in operating telephone companies. They include programming for computers made by all the major computer hardware vendors as well as programming for special-purpose computers designed at Bell Laboratories and built by the Western Electric Company.

Because computer science is still in an early stage of development, no well-formulated theoretical structure exists around which problems can be defined and results organized. "Elegance" is of prime importance, but is not easily defined or described. Reliability and maintainability are important, but they also are neither precisely defined nor easily measured.

No single issue of the B.S.T.J. can characterize all of Bell Laboratories software activities. However, by using the UNIX* operating system as a central theme, it has been possible to assemble a number of related articles that do provide some idea of the importance of computer programming to Bell Laboratories. The original design of the UNIX system was an elegant piece of work done in the research area, and that design has proven useful in many applications. The range of applications described here typifies much of Bell Laboratories software work with the notable omissions of real-time programming for switching control systems and the design of very large data-base systems. Given the growing importance of computers to the Bell System and the growing importance of programming to the use of computers, it is certain that computer programming will continue to grow in importance at Bell Laboratories.

* UNIX is a trademark of Bell Laboratories.

UNIX Time-Sharing System:

Foreword

by M. D. McILROY, E. N. PINSON, and B. A. TAGUE
(Manuscript received March 17, 1978)

Intelligence ... is the faculty of making artificial objects, especially tools to make tools. — Bergson

UNIX is a trademark for a family of computer operating systems developed at Bell Laboratories. Over 300 of these systems, which run on small to large minicomputers, are used in the Bell System for program development, for support of telephone operations, for text processing, and for general-purpose computing; even more have been licensed to outside users. The papers in this issue describe highlights of the UNIX family, some important uses, and some UNIX software tools. They also attempt to convey a feeling for the particular style or outlook on program design that is both manifest in UNIX software and promoted by it.

The UNIX story begins with Ken Thompson's work on a cast-off PDP-7 minicomputer in 1969. He and the others who soon joined him had one overriding objective: to create a computing environment where they themselves could comfortably and effectively pursue their own work—programming research. The result is an operating system of unusual simplicity, generality, and, above all, intelligibility. A distinctive software style has grown upon this base. UNIX software works smoothly together; elaborate computing tasks are typically composed from loosely coupled small parts, often software tools taken off the shelf.

The growth and flowering of UNIX as a highly effective and reliable

time-sharing system are detailed in the prizewinning ACM paper by Ritchie and Thompson that has been updated for this volume. That paper describes the operating system proper and lists the important utility programs that have been adopted by descendant systems as well. There is no more concise summary of the UNIX time-sharing system than the oft-quoted passage from Ritchie and Thompson:

It offers a number of features seldom found even in larger operating systems, including

- (i) A hierarchical file system incorporating demountable volumes,
- (ii) Compatible file, device, and inter-process I/O,
- (iii) The ability to initiate asynchronous processes,
- (iv) System command language selectable on a per-user basis,
- (v) Over 100 subsystems including a dozen languages.

Implementation details are covered in a separate paper by Thompson. Matters of efficiency and design philosophy are considered in a retrospective paper by Ritchie.

The most visible system interface is the "shell," or command language interpreter, through which other programs are called into execution singly or in combination. The shell, described by Bourne, is actually a very high level programming language that talks about programs and files. Particularly noteworthy are its notations for input-output connections. By making it easy to combine programs, the shell fosters small, coherent software modules.

The UNIX system and most software that runs under it are programmed in the general-purpose procedural language C. C provides almost the full capability of popular instruction sets in a setting of structured code, structured data, and modular compilation. C is easy to write and (when well-written) easy to read. The language and the philosophy behind it are covered by Ritchie, Johnson, Lesk, and Kernighan.

Until mid-1977, the UNIX operating system and its variants ran only on computers of the Digital Equipment Corporation PDP-11 family. In an interesting exercise in portability, Johnson and Ritchie exploited the machine-independence of C to move the operating system and the bulk of its software to a quite different Interdata machine. Careful parameterization and some repackaging have made it possible to use largely identical source code for both machines.

Variations

Three papers by Bayer, Lycklama, and Christensen describe

variations on the UNIX operating system that were developed to accommodate real-time processing, microprocessor systems, and laboratory support applications. They were motivated by the desire to retain the benefits of the UNIX system for program development while offering different trade-offs to the user in real-time response, hardware requirements, and resource management for production programs. Many UNIX utilities—especially those useful for writing programs and processing text—will run under any of these variant systems without change.

The MERT operating system (Lycklama and Bayer) provides a generalized kernel that permits extensive interprocess communication and direct user control of peripherals, scheduling, and storage management. Applications with stringent requirements for real-time response, and even different operating systems (in particular, UNIX) can be operated simultaneously under the MERT kernel.

The microprocessor version of the UNIX operating system (Lycklama) and the Satellite Processing System that shares process execution between one big and one tiny machine (Lycklama and Christensen) involve other trade-offs between efficiency and resource requirements. Both also may be looked upon as vehicles for applications in which one wishes to delegate some sticky part of the job—frequently involving real-time demands—to a dedicated machine. The application described later in the issue by Wonsiewicz, Storm, and Sieber is a particularly interesting example involving UNIX, the microprocessor system, and the Satellite Processing System.

Software Tools

Perhaps the most widely used UNIX programs are the utilities for the editing, transformation, analysis, and publication of text of all sorts. Indeed, the text-processing utilities covered by Kernighan, Lesk, and Ossanna were used to produce this issue of the B.S.T.J. Some more unusual applications that become possible where text processors and plenty of text are ready at hand are described by McMahon, Morris, and Cherry.

UNIX utilities are usually thought of as tools—sharply honed programs that help with generic data processing tasks. Tools were often invented to help with the development of UNIX programs and were continually improved by much trial, error, discussion, and redesign, as was the operating system itself. Tools may be used in combination to perform or construct specific applications.

Sophisticated tools to make tools have evolved. The basic typesetting programs `nroff` and `troff` covered by Kernighan, Lesk, and Ossanna help experts define the layouts for classes of documents; the resulting packages exhibit only what is needed for one particular type of document and are easy for nonspecialists to use. Johnson and Lesk describe Yacc and Lex, tools based in formal language theory that systematize the construction of compiler "front ends." Language processors built with the aid of these tools are typically more precisely defined and freer from error than hand-built counterparts.

The UNIX system was originally designed to help build research software. What worked well in a programming laboratory also worked well on modest projects to develop minicomputer-based systems in support of telephone company operations. Such projects are treated in the final group of papers and are more fully introduced by Luderer, Maranzano, and Tague. The strengths of this environment proved equally attractive to large programming projects building applications for large computers with operating systems that were less tractable for program development. The PWB/UNIX extensions discussed by Dolotta, Haight, and Mashey provide such projects with a "front end" for comfortable and effective program development and documentation, together with administrative tools to handle massive projects.

Style

A number of maxims have gained currency among the builders and users of the UNIX system to explain and promote its characteristic style:

- (i) Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features."
- (ii) Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
- (iii) Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.
- (iv) Use tools in preference to unskilled help to lighten a

programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

Illustrations of these maxims are legion:

- (i) Surprising to outsiders is the fact that UNIX compilers produce no listings: printing can be done better and more flexibly by a separate program.
- (ii) Unexpected uses of files abound: programs may be compiled to be run and also typeset to be published in a book from the same text without human intervention; text intended for publication serves as grist for statistical studies of English to help in data compression or cryptography; mailing lists turn into maps. The prevalence of free-format text, even in "data" files, makes the text-processing utilities useful for many strictly data processing functions such as shuffling fields, counting, or collating.
- (iii) The UNIX system and the C language themselves evolved by deliberate steps from early working models that had at most a few man-months invested in them. Both have been fully recoded several times by the same people who designed them, with as much mechanical aid as possible.
- (iv) The use of tools instead of labor is nicely illustrated by typesetting. When a paper needs a new layout for some reason, the typographic conventions for paragraphs, subheadings, etc. are entered in one place, then the paper is run off in the new shape without retyping a single word.

To many, the UNIX systems embody Schumacher's dictum, "Small is beautiful." On the other hand it has been argued by Brooks in *The Mythical Man Month*, for example, that small is unreal; the working of a handful of people doesn't extrapolate to the world of big jobs. We agree only in part, for the present volume demonstrates with unusual force another important factor: intelligently applied computing technology can compress jobs that used to be big to manageable size. The first system had only about 5 man-years' work in it (including operating system, assembler, Fortran, and many other utilities) when it began to be used for Bell System projects. It was, to be sure, a taut package that lacked the gamut of libraries, languages, and support for peripheral equipment typical of a large commercial system. But the base was unusually

pliable and responsive; new facilities usually could be added with much less work than is required by corresponding features in other systems.

The UNIX operating system, the C programming language, and the many tools and techniques developed in this environment are finding extensive use within the Bell System and at universities, government laboratories, and other commercial installations. The style of computing encouraged by this environment is influencing a new generation of programmers and system designers. This, perhaps, is the most exciting part of the UNIX story, for the increased productivity fostered by a friendly environment and quality tools is essential to meet ever-increasing demands for software. UNIX is not the end of the road in operating system innovations, but it has been a significant step that Bell Laboratories people are proud to have originated.

UNIX Time-Sharing System:

UNIX on a Microprocessor

By H. LYCKLAMA

(Manuscript received December 5, 1977)

The decrease in the cost of computer hardware, brought about by the advent of the microprocessor and inexpensive solid-state memory, has brought the personal computer system to reality. Although the cost of software development shows no sign of decreasing soon, the fact that a large amount of software has been developed for the UNIX time-sharing system in the high-level language, C, makes much of this software portable to another processor with rather limited hardware in comparison. A single-user UNIX system has been developed for the DEC LSI-11 microprocessor using 20K words of primary memory and floppy disks for secondary storage. By preserving the user-system interface of the UNIX system, it is possible to run almost all of the standard UNIX languages and subsystems on this single-user version of the UNIX system.*

A background process as well as foreground processes may be run. The file system is "UNIX-like," but has provisions for dealing with contiguous files. Subroutines have been written to interface to the file system on the floppy disks. Asynchronous read/write routines are also available to the user.

The LSI-UNIX system (LSX) has appeal as a stand-alone system for dedicated applications, as well as many potential uses as an intelligent terminal system.

I. INTRODUCTION

The UNIX operating system¹ has enjoyed wide acceptance as a powerful, general-purpose time-sharing system. It supports a large

* UNIX is a trademark of Bell Laboratories.

variety of languages and subsystems. It runs on the Digital Equipment Corporation PDP-11/40, 11/45, and 11/70 computers, all 16-bit word machines that have a memory management unit which makes multiprogramming easy to support. The UNIX system is written in the system programming language, C,² in which most user programs and subsystems are written. Other languages and subsystems supported include Basic, Fortran, Snobol, TMG, and Yacc (a compiler-compiler). The file system is a general hierarchical structure supporting device independence.

With the advent of the DEC LSI-11 microprocessor³ it has become desirable to transport to this machine as much as possible of the software developed for the UNIX system. One of the biggest problems faced is the lack of a memory management unit, which limits the total address space of both system and user to 28K words. The challenge, then, is to reduce the 20K-word, original UNIX operating system to 8K words and yet maintain a useful operating system. This limits the number of device drivers as well as the system functions that can be supported. The secondary storage used is floppy disks (diskettes). The operating system was written in the C language and provides most of the capabilities of the standard UNIX operating system. The system occupies 8K words in the lower part of memory, leaving up to 20K words for a user program. This configuration permits most of the UNIX user programs to run on the LSI-11 microcomputer. The operating system (LSX) allows a background process as well as foreground processes.

The fact that a minimum system can be configured for about \$6000 makes the LSX system an attractive stand-alone system for dedicated applications such as control of special hardware. The system also has appeal as an intelligent terminal and for applications that require a secure and private data base. In fact, this is a personal computer system with almost all the functions of the standard UNIX time-sharing system.

This paper describes some of the objectives of the LSX system as well as some of its more important features. Its capabilities are compared with the powerful UNIX time-sharing system which runs on the PDP-11/40, 11/45, and 11/70 computers,⁴ where appropriate. A summary and some thoughts on future directions are also presented.

II. WHY UNIX ON A MICROPROCESSOR?

Why develop a microprocessor-based UNIX system? The

increasing trend to microprocessors and the proliferation of intelligent terminals make it desirable to harness the UNIX software into an inexpensive microcomputer and give the user a personal computer system. A number of factors must be considered in doing this:

- (i) Cost of hardware.
- (ii) Cost of software.
- (iii) UNIX software base.
- (iv) Size of system.

The hardware costs of a computer system have come down dramatically over the last few years (even over the past few months). This trend is likely to continue in the foreseeable future. Microprocessors on a chip are a reality. The cost of primary memory (e.g., dynamic MOS memory) is decreasing rapidly as 4-kb chips are being replaced by 16-kb chips. A large amount of expertise exists in PDP-11 hardware interfacing. The similarity of the Q-bus of the LSI-11 microcomputer to the UNIBUS of other members of the PDP-11 family of computers makes this expertise available.

Software development costs continue to increase, since the development of new software is so labor-intensive, making it difficult to estimate the cost of writing a particular software application program. Until automatic program writing techniques become better understood and used, this trend is not likely to be turned around any time soon. Thus it becomes imperative to take advantage of as much software that has already been written as possible, including the tremendous amount of software that has already been written to run under the UNIX operating system. The operating system developed for the LSI-11 microcomputer supports most of the UNIX user programs which run under UNIX time-sharing, even though LSX is a single-user system. Thus most of the software for the system is already available, minimizing the cost of software development.

With the advent of some powerful microprocessors, the sizes of some computer systems have shrunk correspondingly. Small secondary storage units (floppy disks) are also becoming increasingly popular. In particular, DEC is marketing the LSI-11 microcomputer, which is a 16-bit word machine with an instruction set compatible with the PDP-11 family of computers. It is conceivable that in the next five years or so the power of a minicomputer system will be available in a microcomputer. It will become possible to allow a user to have a dedicated microcomputer rather than a part of a

minicomputer time-sharing system. LSX is a step in this direction. It will give the user a cost-effective interactive and powerful computer system with a known response time to given requests, since the machine is not time-shared. A dedicated, one-user system can be made available to guarantee "instantaneous" response to requests of a user. There are no unpredictable time-sharing delays to deal with. The system has applications in areas where security is important. A user can gain access to the system only in the room in which the system resides. It is thus possible to limit access to a user's data.

Local text-editing and text-processing features are now available. Other features can be added easily. Interfaces to special I/O equipment on the Q-bus for dedicated experiments can be added. The user then has direct access to this equipment. Using floppy disks as secondary storage gives the user a rather small data base. A link to a larger machine can provide access to a larger data base. Interfaces such as the DLV11 (serial interface) and the DRV-11 (parallel interface) can provide access to other computers.

One of the main benefits of using the UNIX software base is that the C compiler is available for writing application programs in the structured high-level language, C. The use of the shell command interpreter⁵ is also a great asset. A general hierarchical file system is available.

The LSX system has two main areas of application:

- (i) Control of dedicated experiments.
- (ii) Intelligent terminals.

As a dedicated experiment controller, one can interface special I/O equipment to the LSI-11 Q-bus and both support and control the experiment with the same LSX system. The applications as an intelligent terminal are many-fold:

- (i) Development system.
- (ii) General text-processing applications.
- (iii) Form editor.
- (iv) Two-dimensional cursor-controlled text editor.

III. HARDWARE CONSIDERATIONS

The hardware required to build a useful LSX system is minimal. The absolute minimum pieces required are:

- (i) LSI-11 microcomputer (with 4K memory).

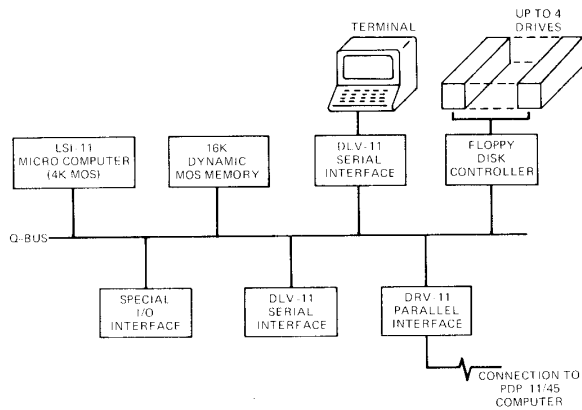


Fig. 1—LSI-11 configuration.

- (ii) 16K memory (e.g., dynamic MOS).
- (iii) EIS chip (extended instruction set).
- (iv) Floppy disk controller with one drive.
- (v) DLV-11 serial interface.
- (vi) Terminal with serial interface.
- (vii) Power supply.
- (viii) Cabinet.

A more flexible and powerful system is shown in Fig. 1. An actual total system is shown in Fig. 2.

The instruction set of the LSI-11 microcomputer is compatible

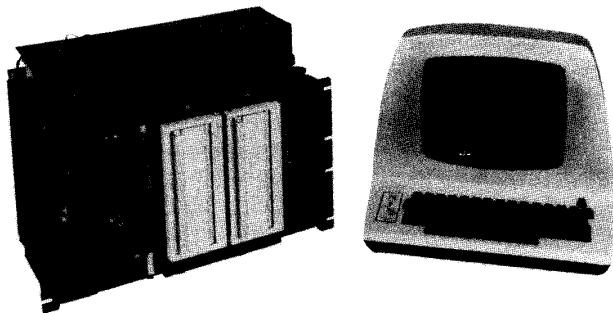


Fig. 2—LSI-11 system.

Table I

Controller	DEC	BTL	AED
Sector size (bytes)	128	512	512
Sectors per track	26	8	16
Number of tracks	77	77	77
Total capacity (bytes)	256256	315392	630784
DMA capability (y/n)	no	yes	yes
Max. transfer rate	6656	24576	49152

with that of the members of the PDP-11 family of computers with the exception of 10 instructions. The missing instructions are provided by means of the EIS chip. These special instructions may be generated by high-level compilers, and it is advantageous not to have to emulate these instructions on the microprocessor. The instructions include the multiply, divide, and multiple shift instructions.

A floppy disk controller with up to 4 drives is shown in Fig. 1. At present, only a few controllers for floppy disks interface to the LSI-11 Q-bus. The typical rotation time of the floppy disks is 360 rpm, i.e., six times per second. All floppy disks have 77 tracks; however, the number of sectors and the size of sectors is variable. The comparative data for the various floppy diskettes are shown in Table I. The maximum transfer rate is quoted in bytes per second. The outside vendor (AED Systems*) supplies dual-density drives for an increase in storage capacity. The DEC drives are IBM-compatible and have less storage capacity. We have chosen to build our own floppy disk controller for some special Bell System requirements.⁶ The advantages of DMA (direct memory access) capabilities are obvious in regard to ease of programming and transfer rate. If IBM format compatibility is important, the throughput and capacity of the system are somewhat diminished.

At least one serial interface card is required to provide a terminal for the user of the system. Provided the terminal uses the standard RS232C interface, most terminals are suitable. For quick editing capabilities, CRT terminals are appropriate. For hard copy, either the common TTY33 or other terminals which run at higher baud rates may be more suitable.

The choice of memory depends on the importance of system size and whether power-fail capabilities are important. Core memory is, of course, nonvolatile, but it takes more logic boards and more

* Advanced Electronics Design, Inc., 440 Potrero Ave., Sunnyvale, California, 94086.

space and is therefore more expensive than dynamic MOS memory. Dynamic MOS memory does not take as much space, is less expensive, and takes less power, but its contents are volatile in case of power dips. Memory boards up to 16K words in size are available* for the LSI-11 microprocessor at a very reasonable price. The memory costs are likely to continue decreasing in the foreseeable future.

Another serial or parallel interface is often useful for connection to a larger machine with a large data base and a complete program development and support system. It is, of course, necessary to use such a connection to bootstrap up a system on the LSI-11 microcomputer. The central machine in this case is used to store all source for the LSX system and to compile the binary object programs required.

The system hardware is flexible enough so that, if necessary, a bus extender may be used to interface special devices to the Q-bus. This provides the ability to add special-purpose hardware that can now be controlled by the LSX system. In Section XI we describe a TV raster scan terminal that was built for editing and graphics applications. Other systems have interfaced special signal-processing equipment to the Q-bus. As DEC provides more of the interfaces to standard I/O peripherals, the applications will no doubt expand.

IV. LSX FILE SYSTEM

The hierarchical file structure of the UNIX system is maintained. The system distinguishes between ordinary files, directories, and special files. Device independence is inherent in the system. Mounted file systems are also supported. Each file system contains its own i-list of i-nodes which contain the file maps. Each i-node contains the size, number of links and the block numbers in the file. Space on disk is divided into 512-byte blocks. In contrast with the UNIX file system, two types of ordinary files are allowed. The "UNIX-type" file i-node contains the block numbers that make up a file. If the file is larger than eight blocks, the numbers in the i-node are pointers to the blocks which contain the block numbers. This requires two accesses to the disk for random file access. LSX recognizes another type of file, the contiguous file, in which the i-node contains a starting block number and the number of consecutive blocks in the file. This requires only one disk access for a random

* Monolithic Memory Systems, Inc.

access to a file, which is important for slow access devices such as floppy disks. Two special commands are provided for dealing with contiguous files; one for allocating space for a file and a second one for moving a file into a contiguous area. The layout of the disk is also crucial for optimum response to commands. By locating directories and i-nodes close to each other, file access is measurably improved over a random distribution on disk.

There is no read/write protection on files. File protection is strictly the user's responsibility. The user is essentially given super-user permissions. Only execute and directory protection is given on files. Group IDs are not implemented. File system space is limited to the capacity of the diskette in use (616 blocks for the Bell Laboratories controller).

V. LSX SYSTEM FEATURES

The LSX operating system is written in the C language, and, as such, bears a strong resemblance to the multi-user UNIX system developed for the PDP-11/40, 11/45, and 11/70 computers. The total system occupies 8K words of memory and has room for only six system buffers. Because the C compiler itself requires up to 12K words of user address space, it is possible to run the C compiler using only 20K words of total memory. It is possible to increase the system size if more capabilities are required in the operating system since the total memory space available to the system and user is actually 28K words. More system buffers could be provided in the system. If the system is kept to 8K words, a 20K-word user program could be run. However, this requires more swap space, which is at a premium.

The system is a single-user system with only one process running at any one time. A process is defined as the execution of an image contained in a file. However, a process may fork up to two levels deep, giving rise to a total of three active foreground processes. The last process forked will run to completion first. More foreground processes can be run, but this requires more swap space on the diskette used for this purpose.

The command interpreter, the shell, is identical to that used in the UNIX system. The file name given as a command is sought in the current directory. If not found, /bin/ is prepended and the /bin directory searched. The /bin directory contains all of the commands generally used. Standard input, output, and diagnostic files are

supported. Redirection of standard I/O is possible. Shell “scripts” are also executable by the command interpreter.

“Pipes” are not supported in the system, but pseudo-pipes are supported in the command shell. Pipes provide an interprocess communication channel in the UNIX time-sharing system. These pseudo-pipes are accomplished by expanding the shell syntax `|` to `> _pf; < _pf`. In other words, a temporary file is used to store the intermediate data passed between the commands. Providing that sufficient disk space exists, the pipe implementation is transparent to the user.

During initialization, the system automatically mounts a user file system on a second diskette if it is desired. The `mount` and `unmount` commands are not available to the user. Thus, a reboot of the system is necessary to mount a new user diskette. The system diskette is normally configured with swap space and temporary file space. User programs and files may reside on the system diskette if a user diskette is not mounted.

The size of memory available and the lack of memory protection (i.e., memory segmentation unit) have put some restrictions on the capabilities of the LSX operating system. However these are not severe in the single-user environment in which the system is run. Profiling is not provided in the system. Timing information only becomes available if a clock interrupt is provided on the LSI-11 event line at 60 times per second. Only one character device driver is allowed at present, as well as only one block device driver. No physical I/O is provided for. There is also no read-ahead on file I/O. Only six system buffers are provided, and the buffering algorithm is much simpler than in the UNIX system. Interactive debugging is not possible, but the planting of break-point traps and post-mortem debugging of a core image is possible. All user programs must be relocated to begin execution at 8K in memory. This required modifications to the UNIX link edit (`ld`) and debugger (`db`) programs. Most other differences between the LSX and the UNIX systems are not perceived by the user.

VI. BACKGROUND PROCESS

It is possible to run a background process on LSX while running a number of foreground processes to get some concurrency out of the system. The background process is run only while the current foreground process is in an input wait state. Two new system calls were added to LSX, `bground` and `kill`, to enable the user to run and

remove a background process. Only one background process is allowed to run and it is not allowed to **fork** another "child" process; however, it may execute another program. The background process either may be compute-bound or may perform some I/O functions, such as outputting to a hard-copy terminal. When the background process is compute-bound, it may take up to 2 seconds to respond to a foreground user's interactive command.

VII. STAND-ALONE ROUTINES

Under LSX, it is possible to run a dedicated program (<20K words) in real time using all the conveniences of the UNIX system calls to communicate with the file system. For programs that require more than 20K words of memory or that require more flexibility than provided by the LSX system, a set of subroutines provide the user with a UNIX-compatible interface to the file system without using the LSX system calls. A user is given more control over the program. Disk I/O issued by the user is buffered using the read-ahead and write-behind features of the standard UNIX system. A much greater number of system buffers are provided than is possible in the LSX system. Eight standard file system interface routines are provided. The arguments required for each routine and the calling sequence are identical to those required by the UNIX system C-interface routines. These include: **read**, **write**, **open**, **close**, **creat**, **sync**, **unlink**, and **seek**. Three unique routines: **saread**, **sawrite**, and **statio** are provided to enable the user to do asynchronous I/O directly into buffers in the user's area rather than into system buffers. These additional routines allow a user to start multiple I/O operations to and from multiple files concurrently, do some computation, and then wait for completion of a particular outstanding I/O transfer at some later time. To provide real-time response in applications that require it, contiguous files may be created by means of an **salloc** routine. The size of the file is specified in blocks. Once created, the file may be grown by means of the **sextend** routine. A **load** program under LSX enables the user to load a stand-alone program that must start execution at location 0 in memory.

VIII. A PROGRAM DEVELOPMENT SYSTEM

One system disk has been configured to contain a fairly complete program development system. The development programs include:

the editor,
the assembler,
the C compiler,
the link editor,
the debugger,
the command interpreter,
and the dump program,

as well as a number of libraries that contain frequently used routines for use by the link editor. It is thus possible to compile, run, and debug application programs completely on-line without access to a larger machine. In a typical application, the contents of the system disk remain quite stable, whereas all user programs are maintained on a permanently mounted user diskette. It is possible to run minimal systems with only one diskette. Although, because of the lack of protection, it is possible to crash the system, in practice, the use of the high-level language C minimizes the number of fatal bugs that actually occur, since the stack frame and program counter are quite well controlled.

In our particular installation, it is often convenient to use the Satellite Processor System⁷ to aid in the running and debugging of new user programs. This is possible since programs running in the LSI-11 satellite microcomputer behave as if they are running on the central machine with access to its file system. This emulates the environment on LSX quite closely. Thus a program may be compiled on a central machine supporting the C compiler, run on the LSI-11 microcomputer, and debugged. When the program has been completely debugged, it is possible to load the program onto the floppy file system using the stand-alone routines described previously and the satellite processor system. This program may then be run under LSX.

IX. TEXT PROCESSING SYSTEM

Another area of application for the LSX system is as a personal computer system for text processing. Files may be prepared using the editor and run off using the UNIX `nroff` command with a hard-copy device. This system disk includes programs such as:

<code>ed</code>	editor
<code>cat</code>	output ASCII files

pr	print ASCII files
od	octal dump files
roff	formatter
nroff	formatter
neqn	mathematical equation formatter

The file transfer program referred to in the previous section enables one to transfer files to or from a machine with a larger data base. Users' files may be maintained on their personal mounted diskettes. If a hard-copy device is attached to the computer as well as to the user's interactive terminal, hard-copy output can be obtained using a background process while another file is edited in the foreground.

X. SUPPORT OF AN LSX SYSTEM

The limited secondary storage capacity available to LSX on floppy disks prevents the mounting of all the system source and user program source code simultaneously. Thus one must be selective as to which programs are mounted at any one time. If a great deal of program development is desirable on LSX, it is often desirable to have a connection to a host machine on which the source code for the application programs can be maintained and compiled. Two means are available to do this. One is to use the Satellite Processor System⁷ and the stand-alone routines described in a previous section as a connection program. This enables one to transfer files (including complete file systems) between the host machine and the satellite processor. The SPS must exist on the host machine and the satellite processor must not be too far distant from the host machine.

A second means of providing support for LSX software is to use a serial line connection such as the DLV-11 between the host machine and the LSI-11 processor. The connection may be either dedicated or dial-up. It requires just five programs, three on the LSX system and two on the host processor. The three programs on LSX include a program to set up the connection to the host machine, i.e., **login** as a user to the host machine, a program to transfer files from the host to LSX, and a third program to transfer files from LSX to the host. On the host machine, the programs include one to transfer a file from the host to the LSX system and vice versa. Complete file systems as well as individual files may be transferred. Checksums are included to ensure error-free transmission.

XI. LSX SYSTEM USES

The LSX system has been put to a number of innovative uses at Bell Laboratories. These include projects that use it as a research tool, for exploratory development in intelligent terminals, and for software support for dedicated applications. LSX is well-suited for the control of an intelligent terminal. As an example, some dual-ported memory has been interfaced to the LSI-11 Q-bus. One port allows direct reading and writing of this memory by the LSI-11 CPU. The other port is used by a microcontroller to display characters on a TV raster scan screen. This enables one to change screen contents "instantaneously." The terminal is suitable for either a two-dimensional text editor or form entry applications. LSX is being used as a vehicle for investigating the future uses of programmable terminals in an office environment for word processing applications.

Other LSX installations are being used to control dedicated hardware configurations. One of the most exciting and in fact the original application for LSX was the software support system for a digital sound synthesizer system. Here the contiguous files supported by LSX are necessary for the real-time application, written as a stand-alone program consisting of a complex multiprocessing system controlling about 100 processes.⁸ The system is capable of existing as a completely stand-alone system and of providing program support on itself.

XII. SUMMARY

The LSX system is currently being used for research in intelligent terminals and in stand-alone dedicated systems. Plans exist to use this system for further research in other areas of Bell Laboratories. Hard-copy features have yet to be incorporated into the system in a clean fashion. Currently, our system is connected to a larger machine using the Satellite Processor System. More general connections to larger machines or possibly to a network of machines has yet to be investigated. The LSX system also has potential uses in multiterminal or cluster control terminal systems where multitasking features are important. These application areas have only been looked at superficially and warrant further investigation.

As a development system, LSX functions quite well. The response to most programs is only a factor of four or so slower than on the conventional minicomputers, due mainly to the slow secondary storage devices used by LSX. Optimization of file storage allocation

on secondary should somewhat improve the response. For instance, the placement of directories close to the i-nodes has improved throughput significantly. The placement of the system swap area needs more investigation as to its effect on throughput.

The advent of large memory boards (64K words) will require the installation of memory mapping to take full advantage of this large address space. This will enable the running of multiple processes without the need for swapping a process out of primary memory and should also improve the response of the system and increase the number of uses to which it can be put.

There is a necessary loss of some functions in the LSX system because of the size of the memory address space available on the LSI-11 computer. However, as a single user system, most functions are still available to the user. As an intelligent terminal system, a microprocessor with all of the UNIX software available is indeed quite a desirable "intelligent" terminal.

XIII. ACKNOWLEDGMENTS

The author is indebted to H. G. Alles for designing and building both the initial PERTEC floppy disk controller and the novel TV terminal. These two pieces of hardware have provided much of the motivation for doing the LSX system in the first place and for doing research in the area of intelligent terminals in particular. Many of the application and support programs described here were written by Eugene W. Stark. John S. Thompson wrote a floppy disk driver for the AED floppy disk controller to facilitate bringing up the LSX system on these disks. The author is grateful to J. C. Swartzwelder and D. R. Weller for their efforts in putting together the first LSI-11 system. M. H. Bradley wrote the initial program to connect the LSX system to a host machine.

REFERENCES

1. D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System," B.S.T.J., this issue, pp. 1905-1929.
2. D. M. Ritchie, S. C. Johnson, M. E. Lesk, and B. W. Kernighan, "UNIX Time-Sharing System: The C Programming Language," B.S.T.J., this issue, pp. 1991-2019.
3. DEC LSI-11 Processor Handbook, 1976.
4. K. Thompson and D. M. Ritchie, *UNIX Programmer's Manual*, Bell Laboratories, May 1975. Sixth Edition
5. S. R. Bourne, "UNIX Time-Sharing System: The UNIX Shell," B.S.T.J., this issue, pp. 1971-1990.
6. H. G. Alles, private communication.

7. H. Lycklama and C. Christensen, "UNIX Time-Sharing System: A Minicomputer Satellite Processor System," B.S.T.J., this issue, pp. 2103-2113.
8. D. L. Bayer, "Real-Time Software for Digital Music Synthesizer," Proc. Second Intl. Conf. of Computer Music, San Diego (October 1977).

This issue of The Bell System Technical Journal was composed, including all tabular and displayed material and final page makeup, using the document preparation software described on pages 2115-2135. It was phototypeset using the troff program, which was written by the late Joseph F. Ossanna, Jr.

THE BELL SYSTEM TECHNICAL JOURNAL is abstracted or indexed by *Abstract Journal in Earthquake Engineering*, *Applied Mechanics Review*, *Applied Science & Technology Index*, *Chemical Abstracts*, *Computer Abstracts*, *Computer & Control Abstracts*, *Current Contents/Engineering, Technology & Applied Sciences*, *Current Contents/Physical & Chemical Sciences*, *Current Index to Statistics*, *Current Papers in Electrical & Electronic Engineering*, *Current Papers on Computers & Control*, *Electrical & Electronic Abstracts*, *Electronics & Communications Abstracts Journal*, *The Engineering Index*, *International Aerospace Abstracts*, *Journal of Current Laser Abstracts*, *Language and Language Behavior Abstracts*, *Mathematical Reviews*, *Metals Abstracts*, *Science Abstracts*, *Science Citation Index*, and *Solid State Abstracts Journal*. Reproductions of the Journal by years are available in microform from University Microfilms, 300 N. Zeeb Road, Ann Arbor, Michigan 48106.

CONTENTS (continued)

G. W. R. Luderer, J. F. Maranzano, and B. A. Tague	The UNIX Operating System as a Base for Applications	2201
B. C. Wonsiewicz, A. R. Storm, and J. D. Sieber	Microcomputer Control of Apparatus, Machinery, and Experiments	2209
A. G. Fraser	Circuit Design Aids	2233
H. D. Rovegno	A Support Environment for MAC-8 Systems	2251
S. P. Pekarich	No. 4 ESS Diagnostic Environment	2265
E. R. Nagelberg and M. A. Pilla	RBCS/RCMAS—Converting to the MERT Operating System	2275
H. Cohen and J. C. Kaufeld, Jr.	The Network Operations Center System	2289
	Contributors to This Issue	2305

