UNIX on a Microprocessor

H. Lycklama

ABSTRACT

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. The cost of software development shows no sign of decreasing soon. However, 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 diskettes. 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. It also has many potential uses as an intelligent terminal system.

March 22,1978

* UNIX is a Trademark of Bell Laboratories.

UNIX on a Microprocessor

H. Lycklama

1. Introduction

The UNIX Operating System (1) has enjoyed a wide acceptance as a powerful general-purpose time-sharing system. It supports a large variety of languages and subsystems. It runs on the Digital Equipment Corporation PDP-11/40, 11/45 and 11/70 computers. These are all 16-bit word machines and have a memory management unit which makes multi-programming easy to support. The UNIX system is written in the system programming language, C (2). In fact most user programs and subsystems are also written in this language. 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 microprocessor (3) it has become desirable to transport as much as possible of the software developed for the UNIX system to this machine. One of the biggest problems faced is the lack of a memory management unit, thus limiting 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 which 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 micro-computer. 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 which require a secure and private data base. In fact, this is a personal computer system with almost all of 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.

2. 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 micro-computer and give the user a personal computer system. There are a number of factors to be considered in doing this:

1. cost of hardware
2. cost of software
3. UNIX software base
4. 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 4K-bit chips are being replaced by 16K-bit chips. There exists a large amount of expertise 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.

The software development costs continue to increase since the development of new software is so labor intensive. It is 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. A tremendous amount of software has been written to run under the UNIX operating system. It seems wise to use as much of this as possible. 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 size of a computer system has shrunk correspondingly. Small secondary storage units (floppy disks) are also becoming increasingly popular. In particular, DEC is marketing the LSI-11 micro-computer which is a 16-bit word machine with an instruction set that is compatible with the PDP-11 family of computers. It is conceivable that in the next 5 years or so the power of a mini-computer 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 mini-computer time-sharing system. LSX is a step in this direction. This 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 DRV11 (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 interpreter (5) is also a great asset. A general hierarchical file system is available.

The LSX system has two main areas of application:

(1) control of dedicated experiments
(2) 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:

(1) development system
(2) general text-processing applications
(3) form editor
(4) two-dimensional cursor-controlled text editor.

3. Hardware Considerations

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

LSI-11 microcomputer (with 4K memory)
16K memory (e.g. dynamic MOS)
EIS chip (extended instruction set)
Floppy disk controller with one drive
DLV11 serial interface
Terminal with serial interface
Power supply
Cabinet.

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

The instruction set of the LSI-11 micro-computer is compatible 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 micro-processor. The instructions include the multiply, divide and multiple shift instructions.

A floppy disk controller with up to 4 drives is shown. At present there are only a few controllers for floppy disks which interface to the LSI-11 Q-bus. The typical rotation time of the floppies is 360 RPM, i.e. six times per second. All floppies have 77 tracks, however the number of sectors and the size of sectors is variable. The comparative data for the various floppy diskettes are as follows:

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

The maximum transfer rate is quoted in bytes per second. The outside vendor (AED Systems (6)) 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 (7). The advantages of DMA (direct memory access) capabilities are obvious as regards 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 non-volatile but it takes more logic boards and more 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 (8) for the LSI-11 micro-processor at a very reasonable price. The memory costs are likely to keep 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 micro-computer. 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 which can now be controlled by the LSX system. In a later section we describe a TV raster scan terminal which was built for editing and graphics applications (8). 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.

4. LSX File System

The hierarchical file structure of the UNIX system is maintained. The system makes a distinction 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 inodes which contain the file maps. Each inode 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 inode contains the block numbers which make up a file. If the file is larger than eight blocks, the numbers in the inode 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 inode contains a starting block number and the number of consecutive blocks in the file. This requires only one disk access for a random 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 inodes 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 id's are not implemented. File system space is limited to the capacity of the diskette in use (616 blocks for the BTL controller).

5. 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 6 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. Re-direction 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 so 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 6 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 transparent to the user.

6. 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 may be either compute-bound or 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 two seconds to respond to a foreground user's interactive command.

7. Stand-Alone Routines

Under LSX it is possible to run a dedicated program (<20K words) in real time using all of the conveniences of the UNIX system calls to communicate with the file system. For programs which require more than 20K words of memory or which require more flexibility than provided by the LSX system, a set of subroutines provide the user 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 of the 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/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 which 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 which must start execution at location 0 in memory.

8. A Program Development System

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

editor
assembler
C compiler
link editor
debugger
command interpreter
dump program

as well as a number of libraries which 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. For minimal systems it is possible to run with only one diskette. Due to the lack of protection, it is possible to crash the system. However in practice, the use of the high-level language C minimizes the number of fatal bugs which 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 System (9) 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.

9. 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:

ed	editor
cat	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/from a machine with a larger data base. A user's files may be maintained on his personal mounted diskette. If a hard-copy device is attached to the computer as well as the user's interactive terminal, hard-copy output can be obtained using a background process while editing another file in the foreground.

10. Support of an LSX System

The limited secondary storage capacity available to LSX on floppies prevents the mounting of all of 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 one desires to do a lot of program development 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. There are two means available to do this. One is to use the Satellite Processor System (9) 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 system 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 DLV11 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.

11. LSX System Uses

The LSX system has been put to a number of innovative uses at Bell Laboratories. These include projects which 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 micro-controller 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 for 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 multi-processing system controlling about 100 processes (10). The system is capable of existing as a completely stand-alone system and providing program support on itself.

12. Summary

The LSX system is currently being used for research in intelligent terminals and in stand-alone dedicated systems. There are plans 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 multi-terminal or cluster control terminal systems where multi-tasking 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 mini-computers. This is due mainly to the slow secondary storage devices used by LSX. Optimization of file storage allocation on secondary should improve response somewhat. For instance, the placement of directories close to the inodes 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. This 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 functionality 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 of the 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.

13. Acknowledgements

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. K.Thompson and D. M. Ritchie, "The UNIX Time-Sharing System", Comm. ACM 17, (July 1974), p365; also this issue.

2. D. M. Ritchie, S. C. Johnson, M. E. Lesk and B. W. Kernighan, "The C Programming Language", this issue.

3. DEC LSI-11 Processor Handbook, 1976.

4. K. Thompson and D. M. Ritchie, "UNIX Programmer's Manual - 6th Edition", May, 1975.

5. S. R. Bourne, "The UNIX Shell - A Command Programming Language", this issue.

6. Advanced Electronics Design, Inc., 440 Potrero Ave., Sunnyvale, CA 94086.

7. H. G. Alles, Private Communication.

8. Monolithic Memory Systems.

9. H. Lycklama and C. Christensen, "A Mini-Computer Satellite Processor System", this issue.

10. D. L. Bayer, "Real-Time Software for Digital Music Synthesizer", Proc. of the Second International Conference of Computer Music, San Diego, October 1977.

FIGURE 1

LSI-11 CONFIGURATION