Bell Laboratories

Cover Sheet for Technical Memorandur

Title - A Structured Operating System for a PDP-11/45

Date - May 6, 1975

TM - 75-1352-4

Other Keywords - UNIX
Multi-Environment
Real-Time

Author(s)	Location and Room	Extension
Lycklama, H.	MH 7C-211	6170
Bayer, D. L.	MH 7C-207	3080

Charging Case - 39394

Filing Case - 39394-11

ABSTRACT

A structured operating system, MERT, consisting of a set of autonomous processes has been designed and implemented on a PDP-11/45 computer in Department 1352. The MERT system is a multi-environment, real-time operating system consisting of a set of basic kernel procedures providing services for the independent processes which support the different operating system environments. A well-developed set of inter-process communication primitives have been implemented, including event flags, message buffers, shared memory and shared files. We believe this provides a good base for providing support for various operating system environments and for providing real-time response for processes. The UNIX time-sharing system has been implemented as one environment on the MERT system. This paper (*) provides an overview of the MERT system.

* - Submitted to COMPCON Conference, September 7-9, 1975

Pages Text	11	Other	5	Total	16
No. Figures	2	No. Tables	0	No. Refs.	4

SEE REVERSE SIDE FOR DISTRIBUTION LIST

DISTRIBUTION
(REFER GEI 13.9-3)

OMPLETE MEMORANDUM TO

ORRESPONDENCE FILES

OFFICIAL FILE COPY
PLUS ONE COPY FOR
EACH ADDITIONAL FILING
CASE REFERENCED

DATE FILE COPY
(FORM E-1328)

10 REFERENCE COPIES

LBERTS,BARBARA A
NDERSON,MRS C M
RDIS,R B
RNOLD,S L
AYER,DOUGLAS L
IREN,MRS IRMA B
LUM,MRS MARION
OYD,GARY D
RANDT,RICHARD B
ROWN,W STANLEY
UCHSBAUM,S J
URROWS,T A
ANADAY,RUDD H
ARDOZA,WAYNE M
ARRAN,J H
ARR,DAVID C
HRISTENSEN,C
LOGSTON,A M
ONDON,J H
OOK,THOMAS J
RANE,RODERICK P
UNNINGHAM,STEPHEN J
UTLER,C CHAPIN
E JAGER,D S
ICK,GEORGE W
OLOTTA,T A
OWD,PATRICK G
DMUNDS,T W
ERRICHIELLO,PHILIP M
FEDER,J
FORTNEY,MRS VIRGINIA J
FRANK,H G
FREENY,S L
ATES,G W
ILLETTE,DEAN
IORDANO,PHILIP P
LASSER,ALAN L
RAVEMAN,R F
HAIGHT,R C
HAMMING,R W
HANNAY,N B
HASKELL,BARRY G
HUPKA,MRS FLORENCE
IVIE,EVAN L
JOHNSON,STEPHEN C
KAPLAN,A E
KAUFMAN,LARRY S
KEVORKIAN,DOUGLAS E
LARSEN,ARTHUR B
LESSEK,PETER V
LIMB,J O

COMPLETE MEMORANDUM TO

LOZIER,JOHN C
LUDERER,GOTTFRIED W R
LYCKLAMA,HEINZ
LYONS,T G
MACHOL,R E JR
MAKER,MS D A
MALTHANER,W A
MARANZANO,JOSEPH F
MASHEY,JOHN R
MC ILROY,M DOUGLAS
MCDONALD,H S
MILLER,S E
MORGAN,S P
NINKE,WILLIAM H
O CONNELL,T F
O NEILL,DENNIS M
OSSANNA,J F JR
PATEL,C K N
PERDUE,R J
PEREZ,MRS CATHERINE D
PERNESKI,A J
PETERSON,RALPH W
PHILLIPS,S J
PILLA,MICHAEL A
PINSON,ELLIOT N
PLAUGER,P J
POPPER,C
+PRIM,ROBERT C
ROBERTS,CHARLES S
ROCHKIND,M J
RODIHAN,MRS PATRICIA A
ROSLER,LAWRENCE
SATZ,L R
SIX,FREDERICK B
SLICHTER,W P
SMITH,D W
SPENCE,NORMAN A
STAMPFEL,JOHN P
STEVENSON,D E
STURMAN,JOEL N
SWANSON,GEORGE K
TAGUE,BERKLEY A
TERRY,M E
TEWKSBURY,S K
THOMPSON,JOHN S
THURSTON,R N
TILLOTSON,L C
UNDERWOOD,R W
VIGGIANO,F A
VOGEL,G C
WALKER,MISS E A
WANDZILAK,P D
WATKINS,G T
WEBB,FRANCIS J
WEHR,L A
WELLER,DAVID R
WHITE,RALPH C JR
WILSON,GEOFFREY A
WOOD,J L
YAMIN,MRS E E
YOUNG,JAMES A
112 NAMES

COVER SHEET ONLY TO

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

ABRAHAM,STUART A
ACKERMAN,A F
AHO,A V
AHRENS,RAINER B
ALCALAY,DAVID
ALLEN,JAMES R
ALLES,HAROLD G
ALMQUIST,R P
AMORY,R W
AMOSS,JOHN J
ANDERSON,M M
ARNOLD,GEORGE W
ARTHURS,EDWARD
ATAL,B S
BAKER,BRENDA A
BALDWIN,GARY L
BALDWIN,GEORGE L
BARTLETT,WADE S
BASEIL,RICHARD J
BAUER,BARBARA T
BAUGH,C R
BECKER,R A
BECKETT,J T
BERGLAND,G DAVID
BERNSTEIN,LAWRENCE
BEYER,JEAN-DAVID
BILOWOS,RICHARD M
BIRCHALL,R H
BLEICHER,EDWIN
BLINN,JAMES C
BLY,JOSEPH A
BODEN,F J
BOHACHEVSKY,I O
BOURNE,STEPHEN R
BOWERS,J L
BOWYER,L RAY
BOYCE,W M
BRAINARD,RALPH C
BREECE,HARRY T III
BREITHAUPT,ALLAN R
BROWN,COLIN W
BUTLETT,D L
BUTZIEN,PAUL E
BYRNE,EDWARD R
BZOWY,D E
CABLE,GORDON G JR
CAMPBELL,J H
CANDY,JAMES C
CASEY,JOSEPH P
CASPERS,MRS BARBARA E
CAVINESS,JOHN D
CHAMBERS,J M
CHAMBERS,MRS B C

COVER SHEET ONLY TO

CHANG,HERBERT Y
CHANG,S-J
CHAPPELL,S G
CHEN,STEPHEN
CHEN,T L
CHERRY,MS L L
CHIANG,T C
CHODROW,MARK M
CHRIST,C W JR
CIRILLO,CARL
CLAYTON,D P
CLIFFORD,ROBERT M
CLOUTIER,J E
COBEN,ROBERT M
COHEN,HARVEY
COLDREN,LARRY A
COLE,LOUIS M
COLE,M O
COLLIER,ROBERT J
COLTON,JOHN R
COPP,DAVID H
COSTANTINO,B B
COSTON,WALTER P
COULTER,J REGINALD
COURTNEY PRATT,J S
CRUME,LARRY L
D STEFAN,D J
DAVIDSON,CHARLES L
DETRANO,MRS M K
DEUTSCH,DAVID N
DICKMAN,B N
DIMMICK,JAMES O
DOMPIERRE,J A
DONOFRIO,L J
DREIZLER,HOWARD K
DRISCOLL,PATRICK J
EDELSON,D
EITELBACH,DAVID L
ELLIOTT,R J
ELY,T C
ESSERMAN,ALAN R
ESTOCK,R G
FABISCH,MICHAEL P
FARGO,GEORGE A
FELS,ALLEN M
FIGLIUZZI,MISS M E
FIORE,MRS RHODA J
FISCHER,H B
FLANAGAN,J L
FLEISCHER,HERBERT I
FLUHR,ZACHARY C
FOUGHT,B T
FOUNTOUKIDIS,A
FOWLER,BRUCE R
FOY,J C
FRANK,MISS A J
FRANK,RUDOLPH J
FRASER,A G
FREEDMAN,M I
FREEMAN,K GLENN
FREEMAN,R DON
FREIDENREICH,MRS B
FROST,H BONNELL

COVER SHEET ONLY TO

FULTON,ALAN W
GARCIA,R F
GAY,FRANCIS A
GEER,EUGENE W JR
GELLINEAU,A C
GELLIS,H S
GEPNER,JAMES R
GEYLING,F T
GIBB,KENNETH R
GILBERT,MRS HINDA S
GIMPEL,JAMES F
GITHENS,JOHN A
GITLIN,RICHARD D
GLUCK,F
GOETZ,FRANK M
GOGUEN,MS NANCY
GOLABEK,MISS R
GOLDSTEIN,A JAY
GORDON,P L
GRAHAM,R L
GRAMPP,F T
GREENBAUM,H J
GREENE,MRS DELTA A
GREENHALGH,H WAIN
GROSS,ARTHUR G
GUERRIERO,JOSEPH R
HAFER,E H
HAGELBARGER,D W
HAGGERTY,J F
HAHN,J R JR
HALFIN,SHLOMO
HALL,ANDREW D JR
HALL,MILTON S JR
HAMILTON,PATRICIA
HANSEN,MRS G J
HARRISON,NEAL T
HARTWELL,WALTER T
HARUTA,K
HASZTO,EDWARD D
HAUSE,A D
HEATH,SIDNEY F III
HELD,RICHARD W
HENIG,MRS FRANCES H
HERGENHAN,C B
HERMAN,KENNETH M
HEROLD,JOHN W
HESS,MILTON S
HOLTMAN,JAMES P
HONIG,W L
HOPPERT,D J
HORNBACH,THOMAS S
HOYT,WILLIAM F
HUDSON,E T
HUMCKE,D J
HUNNICUTT,CHARLES F
IMAGNA,CLYDE P
IPPOLITI,O D
IRVINE,M M
JACKOWSKI,D J
JACOBS,H S
JAMES,DENNIS B
JARVIS,JOHN F
JENKINS,J MICHAEL

...
438 TOTAL

+ NAMED BY AUTHOR   > CITED AS REFERENCE SOURCE

ERCURY SPECIFICATION..

OMPLETE MEMO TO:
135-DPH   13-DIR   11-EXD   15-EXD   16-EXD   127-SUP   135-SUP
UNOS = UNIX/OPERATING SYSTEM

OVER SHEET TO:
135-MTS   9152-MTS   1271   1273   8234
COOS = COMPUTING/OPERATING SYSTEMS/SURVEY PAPERS ONLY

RADY,J E
MH 7B201

TM-75-1352-4
TOTAL PAGES 12

'O GET A COMPLETE COPY:

. BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
. FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
. CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.

PLEASE SEND A COMPLETE COPY TO THE ADDRESS SHOWN ON THE OTHER SIDE
NO ENVELOPE WILL BE NEEDED IF YOU SIMPLY STAPLE THIS COVER SHEET TO THE COMPLETE COPY.
IF COPIES ARE NO LONGER AVAILABLE PLEASE FORWARD THIS REQUEST TO THE CORRESPONDENCE FILES.

Bell Laboratories

subject: A Structured Operating System for a PDP-11/45

date: May 6, 1975

from: H. Lycklama

D. L. Bayer

TM-75-1352-4

Memorandum for File

Introduction

As operating systems become more sophisticated and complex, providing more and more services for the user, they become increasingly difficult to modify and maintain. Fixing a "bug" in some part of the system may very likely introduce another "bug" in a seemingly unrelated section of code. Changing a data structure is likely to have major impact on the total system. It has thus become increasingly apparent over the past years that adhering to the principals of structured modularity (1),(2) is the correct approach to building an operating system. The influence of a process must be confined to an environment which is well protected from the rest of the system and must never affect the state of other environments.

Brinch Hansen (2) implemented a nucleus of a multiprogramming system on an RC 4000 computer using message buffering as the basic means of inter-process communication. Our system uses a different set of message buffering primitives as well as other communication primitives to achieve process synchronization and information transfer in an efficient manner.

The Hardware

The PDP-11/45 computer provides an eight-level hierarchical interrupt structure with priority levels numbered from Ø (lowest) to 7 (highest). Associated with the interrupt structure is the programmed interrupt register which permits the processor to generate interrupts at priorities of one through seven. The programmed interrupt serves as the basic mechanism for driving the system.

The memory management unit provides a separate set of address mapping and access control registers for each of the processor modes: kernel, supervisor and user. Furthermore, each virtual address space can provide separate maps for instruction references (called I-space) and data references (called D-space). Segments may be given read/write protection (3). Each process has access only to its own separate address space.

System Structure

The basic computer hardware resources consist of the actual memory, the CPU and the various I/O devices. The first level (see Fig. 1) of the operating system structure, called the kernel, controls and allocates these resources. The kernel consists of a set of highly privileged procedures and therefore must be very reliable.

The second level of software consists of kernel-mode processes which comprise the various I/O device drivers. Each process at this level has access to a limited number of I-space base registers in the kernel mode, providing a firewall between it and sensitive system data accessible only using D-space mode.

At the third software level are the various operating system supervisors which run in supervisor mode. These processes provide the environments which the user sees and the interface to the basic kernel services.

At the fourth level are the various user procedures which execute in user mode under control of the supervisory environments. The primitives available to the user are provided by the supervisory environments which catch the user traps. Actually the user procedure is merely an extension of the supervisor process. This is the highest level of protection provided by the PDP-11/45 hardware. Higher levels of software such as application packages can achieve protection through software strategies such as running in interpretive mode.

Processes and Segments

A process is a related collection of logical segments which are executed by the processor. A logical segment is a piece of contiguous memory, 32 to 32K 16-bit words, which can grow in increments of 32 words. A name may be associated with a segment to provide a label for sharing it between independent processes. This name is derived from the file system and is thus globally known to all processes.

The capabilities of a process are defined by the mode of the processor while executing the process segments. A kernel process, running in kernel mode, is limited to 12K words, is locked in memory and has direct access to the I/O registers. Kernel processes can execute at processor priorities of three to seven and are driven by both software and external interrupts. These processes can respond to interrupts within 100 microseconds.

All processes which do not run in kernel mode are supervisor processes. These processes all execute at processor priorities one and zero and are scheduled by the kernel scheduler process. Associated with each supervisor process is a special read-only segment (in supervisor mode) called the process control block (PCB). The PCB contains space to save the state of the process and a segment table which describes the virtual address space of the process. The process can manipulate the segment table by trapping to the kernel. Supervisor processes can be either swappable or non-swap depending upon real time constraints. The maximum size of a process is limited by the addressing capability of the hardware to 124K words. Real time response attainable by supervisor processes is of the order of several milliseconds.

Process Communication

A number of methods are available to the processes for communicating with each other. These include:

event flags
messages
shared memory
files

These methods can be used to guarantee the synchronization of parallel, cooperating processes and for maintaining the modularity of these processes.

The most basic communication mechanism transmits one bit event flags:

event(procid, event)

which will cause the process procid to receive an interrupt and the word event, which contains one or more event flags. The receiving process can inhibit event interrupts by selectively enabling particular events.

A higher-bandwidth communication path is provided by the message sending and receiving mechanisms in the kernel. A common pool of message buffers (multiple of 16 words) and a message queue is provided for each process. See Figure 2 for the message format. Communication between all processes can be achieved by means of the following kernel primitives:

send message(message-buffer)

receive message(message-buffer)

receive message type(message-buffer)

Send message copies the message from the process message buffer into the next available kernel message buffer and places it on the input message queue of the intended receiver and sends a message event to the receiver. Receive message copies a message into the process message buffer if there is a message on its input queue. If the queue is empty, the process decides whether to continue or roadblock while waiting for a message. A process may also look for a particular type of message on its input message queue by means of the receive message type primitive. This enables a process to synchronize the processing of its acknowledgement messages. The transmission of a message almost always results in the receipt of an acknowledgement or answer message.

Processes have the capability to share segments. For example, a segment may be shared as a common pool of I/O buffers for the use of co-operating processes. This sharing mechanism is used in the implementation of a time-sharing supervisor in the system. Shared segments are also be used to transfer information between two co-operating processes.

The use of files for inter-process communication is common in many operating systems. In our case named files are passed between communicating processes. We also provide the mechanism to lock a file, i.e. make it inaccessible to other processes. The existence of the locked file is a means of inter-process communication. The contents of a file may of course also be shared by a number of processes.

The Kernel

The kernel consists of a process dispatcher, a trap handler, and routines which implement the system primitives.

The process dispatcher fields all interrupts, sets up the appropriate kernel mode process, and transfers control to it.

The trap handler fields all traps and faults and, in most cases, transfers control to a trap handling routine in the process which caused the trap or fault. For the purposes of debugging, the "break point trap" executed from supervisor or kernel mode will cause an image of the process to be written in a file and the process to be terminated. This feature has proven to be an invaluable tool for debugging kernel processes which control I/O devices and new versions of our time sharing supervisor.

The kernel primitives deal primarily with four logical functions:

1) The sending and receiving of messages and events.
2) The allocation and manipulation of segments which includes such operations as locking for I/O, growing, and sharing segments.
3) The manipulation of a process virtual address space. This includes specification of access permissions, starting virtual address, and growth direction of segments.
4) The manipulation of process attributes which influence scheduling. Included here would be altering the process priority and making the segments of a process unswappable.

Closely associated with the kernel are the memory management and scheduler processes. These two processes are special in the sense that they have access to kernel D-space and reside in the kernel segments. In all other respects they follow the discipline established for kernel processes.

The memory manager process communicates with the rest of the system via messages and is capable of handling three types of requests:

1) Setting the segments of a process into the active state, making space by swapping or shifting other segments if necessary.

2) Loading and locking a segment contiguous with other locked segments to reduce memory fragmentation.

3) Deactivating the segments of a process.

The scheduler process is responsible for scheduling all supervisor mode processes. The scheduler utilizes both time-sliced, round robin and preemptive priority scheduling techniques. The main responsibility of the scheduler is to select the next process to be executed. The actual loading of the process is accomplished by the memory manager.

File System

The system has the capability to provide such diverse environments as those suitable for real-time applications, database query systems and time-sharing systems. The types of files normally provided by operating systems cover the range from linked files to large and contiguous files, although these types rarely all appear on one system. Large and contiguous files in particular are appropriate for real-time applications and database query systems, and lend themselves to more sophisticated file I/O capabilities.

The file manager process provides the capabilities for all of these file types using a single type of file map entry. A file map entry describes the file layout by a series of 2-word extents consisting of starting block number and number of consecutive blocks. This format enables the user to allocate secondary storage space for a large contiguous file. For time-sharing applications the file may consist of a number of small extents. The maximum file size which can currently be accommodated under this scheme is 850,000,000 bytes.

The file manager process is implemented as a separate process controlling all accesses to files and is the only process which may alter the file map entries. Other processes communicate with the file manager entirely by means of messages. The messages to the file manager are handled by a number of tasks running in parallel, one on behalf of each message.

Status

The Operating System as discussed above has been implemented on a PDP-11/45 computer with 80K words of memory. The size of the kernel is 8K including system tables. Two basic system processes are started up with the system, a process manager and a file manager. A time-sharing supervisor, logically equivalent to UNIX (4), has been implemented as an environment, requiring about 11K of memory. At the current state of implementation of the system, the system overhead introduced by the message traffic and context changing reduces the throughput of the UNIX time-sharing processes by about 5 percent.

Development of new and existing supervisors is being done using this time-sharing system for which many user application programs have been written over the years. One real-time supervisor currently being developed is an experimental telephone office common control used for call processing. This process is locked in memory and runs at high priority upon receiving a message or an event. This process may be loaded dynamically and run under control of the time-sharing supervisor. It is debugged by planting break-point traps and dumping out a memory image of the process on a file. In fact many of the kernel device drivers have been debugged in this manner. This allows the system to remain operational while some very sensitive code is being debugged. The debugging facilities allow a user to examine the state of any process in detail either while running or after being dumped.

The flexibility of the system is perhaps best shown by the ability to run and debug a new version of the time-sharing supervisor using the current version of that supervisor. This enables the system designer to test out new features of the time-sharing supervisor without running the risk of disrupting other users. The debugged version of the supervisor may then be installed without taking down the system. The structure of the operating system continues to provide a base for doing further operating systems research.

Acknowledgments

Some of the concepts incorporated in the basic kernel were developed in a previous design and implementation of an operating system kernel by Mr. C. S. Roberts and one of the authors (H. Lycklama). The authors are pleased to acknowledge Mr. C. S. Roberts for many fruitful discussions during the design stage of the current operating system.

[illegible]

H. Lycklama

[illegible]

D. L. Bayer

MH-1352-HL-JER

Att.
References
Figures

References

(1) Dijkstra, E.W., The Structure of the ´THE´ Multi-Programming System. Comm. ACM 11, (May 1968), p341.

(2) Brinch Hansen, P., The Nucleus of a Multi-Programming System. Comm. ACM 13, (April 197ø), p238.

(3) PDP -11/45 Processor Handbook, Digital Equipment Corporation, Maynard MA, 1971.

(4) Thompson, K. and Ritchie, D.M., The UNIX Time-Sharing System. Comm. ACM 17, (July 1974), p365.

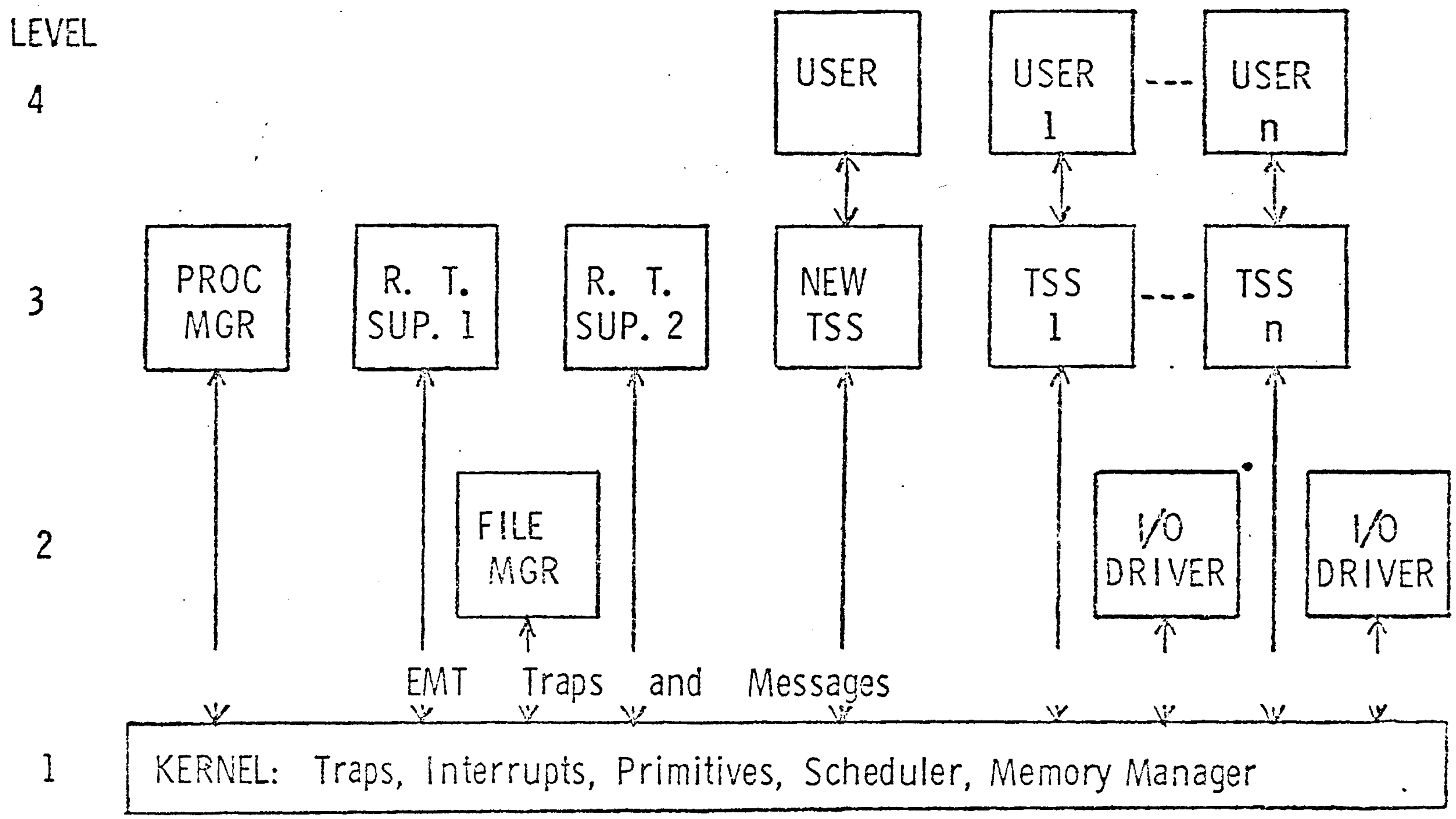

Figure 1 System Structure

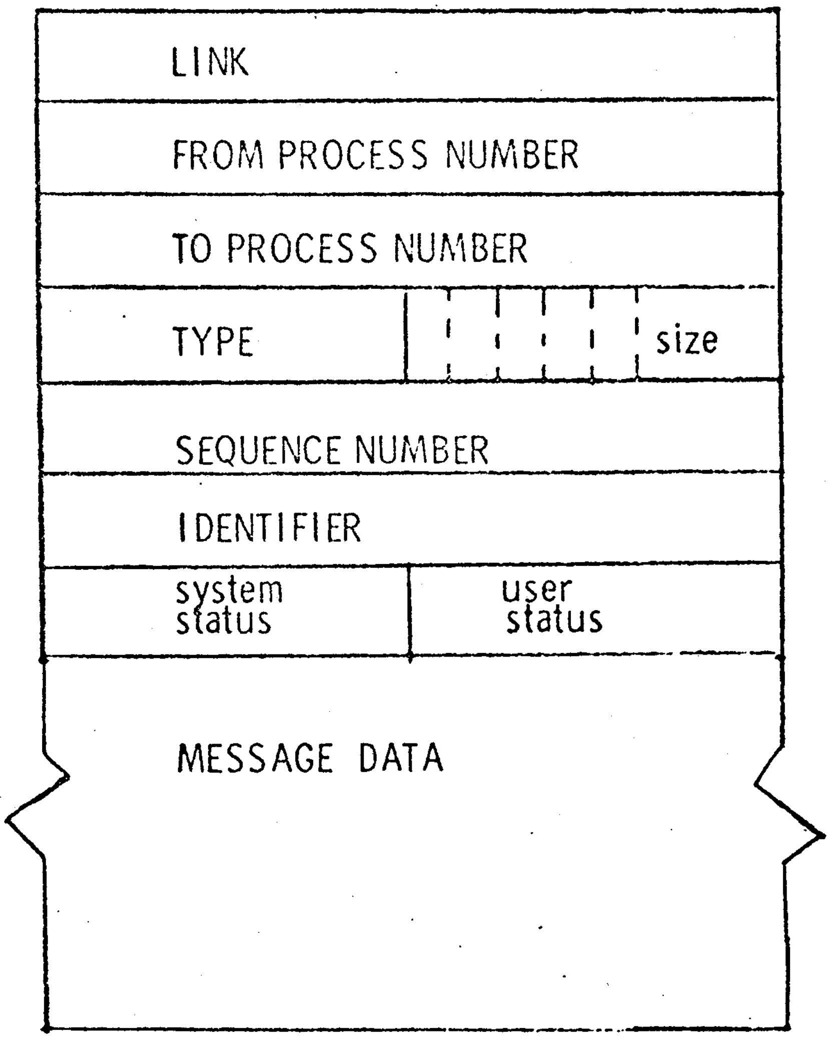

FIGURE 2 Message format