THE MITRE CORPORATION

BEDFORD, MASS.

MEMORANDUM

TO: E. L. Burke

DATE: 27 February 1975

FROM: K. J. Biba

MEMO NO.: D73-M799

SUBJECT: Multics Design Note #13 - Kernel Levels of Abstraction

COPIES TO: Distribution

Multics Design Note #13, Kernel Levels of Abstraction, is attached.

R. J. Biba

K. J. Biba
Intelligence and
Information Systems

KJB:jkl

Attachment

Distribution:

S. R. Ames, Jr.
D. E. Bell
E. H. Bensley
K. J. Biba
E. L. Burke
C. S. Chandersekaran
M. Gasser
C. D. Jordan
L. J. LaPadula
S. B. Lipner
J. K. Millen
R. D. Rhode
W. L. Schiller
D. F. Stork
J. C. C. White

Attachment 1

Context of Design Note

This design note documents a revision to the Multics' kernel decomposition proposed in Multics Design Note #2. Recent work on the Multics' specification has resulted in a reorganization and compression of the previous decomposition. Seven (7) levels are now suggested where Level 0 represents the hardware and Level 6 represents the security kernel perimeter.

The presentation of these levels is in a "bottom up" manner. While the top level (Level 6) is defined "first", since it represents the desired and observable (neglecting timing) behavior of the kernel, the "lower" levels may be defined in any convenient order. The presentation viewpoint illustrates how the kernel perimeter functionality may be incrementally constructed, in a well-formed manner, from the hardware base.

Level 0: HIS 68/80 Hardware

Level 0, as in the previous decomposition, specifies the kernel's hardware base. However, the hardware considered will be the HIS 68/80 rather than the HIS 6180. The major difference, for the specification, appears to be the existence of per-processor cache store. The encacheability of segments is determined by a 1-bit field in each segment descriptor (SDW). It is anticipated that the kernel's read only data bases (particularly instructions) are encacheable, while read/write data bases (particularly global data bases as the APT and AST) are not.

Level 1: Primary Memory Management

Level 1 defines a segmented memory environment utilizing a single (physical) level of memory. Provision is made for two virtual → real address mapping algorithms: for paged and unpaged segments. The necessary constructs for the management of this level of memory are provided. Primary memory is considered as a set of memory frames (each 1024_{10} words) which may be assigned (allocated) to a segment. Primitives are provided to add and remove primary memory frames and segments from their respective free (unallocated) pools. Refer to Multics Design Note #10 for a preliminary specification of this level.

Level 2: Processor Management

Level 2 defines a set of primitive processes and the mechanism by which they are bound to (physical) processors. A resource pool of processors is defined, and primitives for the addition and deletion of processors from this pool are also defined. Primitives for the synchronization of these processes are provided as well as primitives for their creation and deletion. Above this level, these processes will constitute a "virtual processor" resource pool upon which other processes (specifically user processes) may be defined and allocated. This abstraction is similar to a virtual processor implementation recently proposed by D. Reed at MIT.

Level 3: Multilevel Storage Management

Level 3 defines resource pools and allocation primitives to provide for multiple levels of physical storage. Pools of pageframe resources are defined both for secondary (probably bulk core) and tertiary (probably disk) physical storage. The concept of segments whose (physical) storage is not (entirely) in primary memory is defined. This level defines sufficient primitives such that pagefault handlers can be constructed.

Level 4: Active Segment Management

Level 4 defines the concept of inactive segments: segments whose attributes are not currently in primary memory. A resource pool of active segment names (all of whose attributes are in primary memory) and resource management primitives are defined. The concept of a directory as a set of inactive segments (and their attributes), of which some may be themselves directories, (ordered as a n-ary tree) is defined. This concept allows the addressing of the attributes of an inactive segment (including its associated data) by an m-tuple, where each element, k, is a local name within the k-1th element denoting an attribute set of an inactive segment. A segment termed the root is an implied zero-th element. All elements k ($o \leq k < m$) must be directories. This level provides sufficient mechanisms so that segment faults may be resolved (through the activation of the segment labeled by an m-tuple).

Level 5: SFEP Interface

Level 5 defines the inter-computer communications protocol to be used between the kernels of Secure Front End Processors and Multics. Sufficient information and operations are defined so that SFEP (Multics) interrupt handlers can be constructed.

Level 6: Security Kernel Perimeter

Level 6 defines the user observable behavior of the Multics security kernel. Multics Design Notes #6, #7, and #11 define the component specifications for external I/O, process control, and storage control.