File -
Project
Guardian

PROJECT GUARDIAN
TECHNICAL COORDINATION LETTER

Date: 27 January 1976

To: Contracting Officer
HQ ESD/MCP
Hanscom Air Force Base
Bedford, Mass. 01731

TCL No: 14

Contract No: F19628-74-C-0193

Attention: C. E. Fenton, Captain, USAF

Subject: Preliminary Modularization of Multics - January 26, 1976

The attached technical note provides the above listed report as prepared by R. Feiertag.

Table I and II have been provided to list the "Hierarchical Modularization for Multics" and "Comparison of Multics Hierarchy to SRI Hierarchy", respectively. A bibliography has also been provided.

If there are any questions, please contact the undersigned or Mr. N. Adleman in our Cambridge, Massachusetts office.

Very truly yours,

HONEYWELL INFORMATION SYSTEMS INC.

R L Carlson

R. L. Carlson
Contract Specialist

RLC/meb
Attachment

cc: ESD/MCI (5)
MITRE-D73, Mr. E. Burke (5)
RADC/ISM (3)
NSA/R14 (3)
AFDSC/XMS (2)
JTSA (5)

01/26/76

Preliminary Modularization of Multics
R. Feiertag

The following pages describe a preliminary hierarchical modularization for Multics. Although Multics is fairly well modularized, the modules are not organized hierarchically. The hierarchicalization is desirable in order to permit the proof methodology [2] to be applied to Multics.

The organization described below represents an attempt to formulate a hierarchically structured Multics with minimum change to the existing Multics. The design proposed is, therefore, not as neat as it could be if one were starting from scratch. Some areas present especially difficult problems that will be discussed later in the description.

At this early stage of design, each level is named with a character string rather than a number. This is because the levels will change a great deal and numbers would only be confusing.

The Design

The modularization of the system is summarized in Table 1. Although there appears to be a large number of levels, it is expected that many of these levels will be combined in the final design. They are presented separately here so that no important functions are hidden inside a level. Each level will be discussed starting with the lowest level.

Directly Addressable Memory (dam)

This level consists of random-access memory addressable directly by the processes. The operations are read, write, and write-conditional. The latter operation is necessary for test and set instructions. This level is entirely hardware.

Usual Instruction Set (uis)

These operations are simply the standard arithmetic, logical, transfer, and conditional transfer instructions of the processor. This level does not include special system instructions such as those that load the descriptor base register, or set or mask interrupt cells. This level is entirely hardware.

Primitive Segments (pseg)

In Multics all addressing (with the exception of a handful of instructions in the very early bootstrap stages) is done in terms of segments. Therefore, some primitive form of segmentation is necessary at the lowest software level. This level provides for segments which are contiguous and resident in primary memory (i.e. non-paged segments). This level must, therefore, manage descriptor segments. The operations of the level will include read and write for segments. Since each Multics address space contains only one descriptor segment, this level must allow higher levels (namely sseg and seg) to access some descriptors; however, the descriptors which higher levels are allowed to access must be carefully controlled. One fundamental problem with the implementation of this level in Multics is that descriptor segments are paged and the first page of the descriptor segment for a process is made primary memory resident only when the process is running. This makes the implementation of this level very tricky and difficult since both processes and pages are several levels higher in the hierarchy. Much thought will be needed about this level.

In addition to providing primitive segmentation, this level will also provide primitive protection. This is done by setting the ring number fields of the descriptor segment entries for all primitive segments to 0. This assures that nonsupervisor programs cannot access the primitive segments. Access control within ring 0 will have to be accomplished by other means (perhaps by suitable language restrictions). This level is implemented as both hardware and software.

Interrupts (int)

This level includes the Multics interrupt and fault mechanisms. These are the means by which processors and other devices may interrupt processors. The Multics interrupt mechanism is somewhat complex and will probably take some effort to specify properly. This level will also include some of the low-level software for handling interrupts and faults such as the fault interceptor and interrupt interceptor modules.

Primitive Input/Output (pio)

The input/output hardware for future Multics has not yet been precisely determined, however, it should be specified at about this level.

Low-Level Traffic Control (lltc)

This level corresponds roughly to level 2 of the SRI secure operating system [1]. It provides a fixed number of virtual processes independent of the number of physical processors. It provides some kind of primitive wait and notify mechanism for these low-level processes. Hopefully all interrupts, with the possible exception of the quit and stop interrupts, will be invisible above this level. Interrupts will be converted to notifies. This level is software.

Low-Level Input/Output (llio)

This level contains the software for the secondary memories (e.g. bulk store and disk). It provides sufficient I/O facilities to perform paging.

The next five levels describe the Multics storage system. The storage system could be described as a single level, but the level would be so large in implementation as to be virtually unverifiable. The breakdown into five levels is an attempt to find a reasonable modularization that will not necessitate rewriting the entire storage system. To attain this modularization, several hard problems will have to be solved (e.g. quota) and these levels will probably be the most difficult in the system to design.

Paging (pg)

This level implements paging. It is responsible for allocating and removing pages in primary and secondary memory and maintaining file maps and page table entries. This level is implemented as both hardware and software.

System Segments (sseg)

This level has the same purpose as level 3 of the SRI secure operating system. It provides segments which are used by higher levels of the supervisor, but not available to users. This allows higher levels of the supervisor to be paged. This level is both software and hardware.

Segment Access (sacc)

This level is responsible for maintaining the access attributes of a segment, namely ring brackets, access control lists, access isolation mechanism information, etc. These functions form a separate level because a sizeable amount of code is involved and it isolates the security policy, with respect to segments, in one level. This level is software.

Segments (seg)

This level implements user visible segments. The operations of the level include read, write, create, delete, etc. on segments. This level will be responsible for maintaining much of the information currently kept in the AST, KST (not including reference names), and descriptor segments. This level is both software and hardware.

Directories (dir)
This level maintains directories. The operations include creating, deleting, adding, removing, and modifying entries, and searching. This level is implemented as software.

High Level Traffic Control (hltc)
This level provides user processes and is analogous to level 10 of the SRI operating system. User processes are implemented in terms of low level processes. Processes can be created, deleted, blocked, and awakened. This level is implemented as software.

Input/Output (io)
This level includes the software necessary to maintain the remaining I/O devices (i.e. those not reserved for the storage system). This level is expected to be software and hardware.

The description of the levels is at this time necessarily vague. Some of the harder problems of the modularization, mainly the levels of the storage system (dir, segm, sacc, sseg, and pg) and the int and pseg levels, need much more investigation.

Several important functions of Multics have not been discussed in the above design. These are reconfiguration; error detection, correction and reporting; initialization and shutdown; salvaging; BOS; and accounting and resource management. The functions of reconfiguration, error handling,, initialization and shutdown, and accounting and resource management are largely distributed through many levels of the operating system with each level being responsible for these functions as it applies to the level. Each of these functions will require some kind of central facility which will either be a separate level or included in some other level. The salvager and BOS need to be handled separately since they do not execute as part of Multics, but they still need to be verified since they have access to data used by Multics.

It is interesting to compare the modularization described above with the SRI operating system. The comparison is summarized in Table 2. The I/O in the Multics design has no correspondence in the SRI design because the I/O part of the SRI design has not yet been formulated. The significant differences in the design are due to the different access mechanisms for the two systems, access lists for Multics and capabilities for SRI. Levels 8 and 9 of the SRI system have no correspondence in Multics because, in Multics, dynamic linking is performed outside the supervisor. There are no extended objects in Multics so there is no equivalent to level 5.

Table 1
Hierarchical Modularization for Multics

Level	Brief Description
io	user visible input/output facilities
hltc	user processes
dir	directories
seg	segments
sacc	segment access control
sseg	supervisor only segments
pg	paging
llio	input/output facilities for storage system devices
lltc	system processes
int	interrupts and faults
pseg	primitive (primary memory resident) segments
pio	input/output hardware
uis	usual instruction set
dam	directly addressable linear memory

Table 2
Comparison of Multics Hierarchy to SRI Hierarchy

Multics	SRI
io	no correspondence
hltc	level 10
dir	level 6
seg	level 4
sacc	no correspondence
sseg	level 3
pg	level 3
llio	no correspondence
lltc	level 2
int	level 0
pseg	no correspondence
pio	no correspondence
uis	level 1
dam	level 0

Bibliography

[1] Neumann, P.G., Robinson, L., Levitt, K.N., Boyer, R.S., Saxena, A.R., A Provably Secure Operating System, Stanford Research Institute, Menlo Park, Ca., June, 1975

[2] Robinson, L., Levitt, K.N., Neumann, P.G., Saxena, A.K., On Attaining Reliable Software for a Secure Operating System, Proceedings International Conference on Reliable Software, SIGPLAN NOTICES, vol. 10, no. 6, June 1975, pp. 267-284. A revised and extended version is being published under the title, A Formal Methodology for the Design of Operating System Software, in R. T. Yeh (ed), CURRENT TRENDS IN PROGRAMMING METHODOLOGY, vol. 1 (Prentice-Hall, 1976).