To: MTB Distribution

From: M. D. Schroeder

Date: November 28, 1973

Subject: Facilitating modifications to ring 0 gates

From time to time an improvement to Multics is suggested which requires changing the call interface to ring 0. When implementing such changes, it is almost always important to maintain upward compatibility. The addition of gate entry points (gates) representing new functions usually is easy to do in an upward compatible way. Upward compatibility can also be maintained when replacing gates with new ones implementing similar functions, at the cost of leaving the replaced gates in the system for a while, either unchanged or altered to translate requests and pass control on to their replacements within ring 0. When functions provided by existing gates are to be moved out of the supervisor to higher numbered rings, however, an upward compatible implementation is usually not possible. The problem is that all ring 0 gates are collected in a few gate segments that execute in ring 0. Because the code of these gates executes in ring 0 it is not possible for that code to routinely pass control to an outer ring replacement. Because all gates are collected in a few large gate segments, directly making a gate become an outer ring procedure implies doing the same for all other gates in the same gate segment, for they are all "known" by the same segment name.

The difficulty in providing outer ring replacements for supervisor functions is very unfortunate, for doing so reduces the size and complexity of the supervisor, making it more maintainable, modifiable, and potentially securable. It is also a difficulty faced by several changes to Multics currently being considered. Phil Janson's proposed user ring dynamic linker, Bob Mabee's proposed filter on supervisor meters, and Dick Snyder's proposed TTYDIM/355 software replacement all will move certain functions out of ring 0.

Solutions

One solution to this problem is making the present gate segments be transfer vectors that execute in the ring from which they are called, e.g., have ring brackets 1,5,5. The present names for gate segments would be applied to the transfer vectors, and each actual gate segment would be given another name. In cases where functions continue to be implemented in the supervisor, the corresponding transfer vector entry points would simply transfer control to the actual gates. Moving a function out of ring 0 is simply implemented by changing the appropriate transfer vector entry points to pass control to the non-supervisor replacement procedures. This transfer is possible since the code in the vectors executes in the ring from which it is called. After all programs are converted to call a gate replacement procedure directly, the unneeded transfer vector entry could be eliminated.

Once making supervisor gate segments be user ring transfer vectors is done, two possibilities present themselves. The first is to treat the transfer vectors as the published system interface. The actual gate segments, with new names, would not be described in the MPM, and users would be warned that the names and functions of the contained gate entry points may change. Thus, users would be discouraged from calling them directly. From then on, changes in the actual supervisor interface could be implemented easily as described above. This first possibility can be realized without changing the current published (in the MPM) description of the system interface.

The first possibility, however, has a long range defect. While user code could be effectively discouraged from directly invoking the actual ring 0 gates, it is inevitable that system-provided procedures executing in outer rings would be produced that do directly invoke these gates. Once this begins to happen, removal of functions from ring 0 is again effectively discouraged by the need to track-down and change all system-provided procedures that directly call the actual gates. In other words, this first possibility allows the current problem to regenerate itself as time goes on.

The second possibility eliminates the chance that the problem will regenerate itself. It is to treat the transfer vectors as an interim step done for reasons of upward compatibility. Once this step is taken, the names of the actual supervisor gate segments would be changed so that each gate entry point was called through a distinct segment name. (With the first possibility, the actual gate segments could continue to have a single segment name.) For example, all gate entry points currently named "<gate_segment_name>$<entry_name>" could become "<entry_name>_$<entry_name>_". These name changes could be accomplished by adding the names of all contained gate entry points as names of the containing gate segment. Once the renaming is done so that each gate entry point has a distinct segment name, then the published description of the supervisor interface would be changed to use the new names, e.g., "hcs_$terminate" would become "terminate_$terminate_" or simply "terminate_". All users would begin using the new names. Old programs would continue to work via the transfer vectors. Eventually, when all programs had been converted, transfer vectors could be eliminated. This second possibility solves the same problem as the first in a different way. Because each gate is associated with a different segment name, the gate can be converted to a non-supervisor procedure directly (or even a ring 1 gate) without affecting any other supervisor gates (even those in the same gate segment), for they are known by different segment names. If the second possibility is adopted, it is important to do so soon, so that the first formally released description of the standard Multics system can specify the new gate names.

Costs

The cost of the preferred solution appears small for the benefit derived. While the transfer vector exists an extra instruction is executed and possibly a page fault is generated when a call is made through a transfer vector entry point, and an extra linkage fault is generated the first time a particular supervisor gate is used within a ring. (The transfer vector could consist of "tra *+n,*" instructions executing in its linkage section, to eliminate a "getlp" macro and another possible page fault.) This portion of the cost is eliminated as the transfer vector falls into disuse and is eliminated. Using distinct segment names for gates causes one segment name per used supervisor gate to be recorded in the KST, rather than the current one name per gate segment. More names would increase KST search time somewhat, but measurements are necessary to see how large an effect this would be.