Published: 06/26/67
(Supersedes: BT.3.02, 06/16/67)

Identification

Reserver implementation plan
J. H. Saltzer

Purpose

This section describes the plan of implementation of the reserver through the first four phases of Multics and beyond.

Phase One.

The reserver will not be called, and not even a dummy module is needed.

Phase Two.

A dummy reserver will be provided. All calls to segments reserve and release will be implemented as a return only, with no status argument set. In fact, only the entries should be called in phase two, in response to certain I/O system disposal options. Allocation calls are handled as follows:

alloc$type (allocation from a pool of resources) is illegal, and will signal alloc_error. The phase two I/O system is restricted in such a way that it should never make this call.

alloc$resource and de_alloc$resource will, without asking any questions, perform the privileged call to the resource assignment module to assign or unassign the requested resource to the calling user. They will then return. It is expected that phase two users will be friendly enough not to fight over allocatable resources.

de_alloc$all will return only. It is expected that phase two users will clean up after themselves, and special cleanup at logout time is not necessary. (This may be the first reserved entry implemented on the way to phase three.)

Phase Three.

In phase three, a primitive reserver capable of allocations only is implemented. This reserver handles reserve and release calls as in phase two. It is capable of allocations from pools of resources and of rejecting requested allocations for devices already allocated to another user.

Phase Four.

The first real reservation-making reserver appears. It can handle all calls except reserve$group. Because reserve$group is not implemented, the reserver strategy is greatly simplified. Each reservable resource has a table of the type described in BT.3.00; no simulation is required to test for the availability of a requested reservation.

A simple reservation-making command is implemented to go along with the phase four reserver.

After Phase Four.

The full reserver described in BT.3.00 is implemented. A sophisticated set of reserver commands is also provided to interface with the full reserver.