TO:	F. J. Corbató	J. H. Saltzer ✓
	C. T. Clingen	M. D. Schroeder
	J. W. Gintell	V. L. Voydock
	N. I. Morris	S. H. Webber

FROM: R. J. Feiertag

DATE: February 23, 1971

SUBJECT: Segment Attributes

Recent events in Multics development have brought to light several issues revolving around segment attributes, i.e., that information about a segment that is kept in the directory branch associated with the segment. In this memo I will attempt to summarize these issues in order that everyone may gain a better overall picture of where the problems are and how they interact. I will treat some issues more quickly than others, however, the depth of treatment reflects my opinion of the awareness of the readers of the issues and not the relative importance of the issues.

There is the question of how directory control determines if a process may read or delete a particular segment attribute. At present such permission may be controlled by the process's access on the directory, its access on the segment, or a combination of the two depending upon the attribute in question. An example of current interest is the segment name. Permission to modify or add a name is derived from access to the directory in which the name resides when it possibly should be derived from access to the segment itself as well as the directory (see Webber's memo of January 11). Each segment attribute should be reexamined to determine how the various permissions associated with it should be derived.

Besides the question of whether permission to read or alter segment attributes should come from the directory or segment there is the question of what access attributes should be granted. At present there are only four types of access attributes, rewa. There is the possiblity that the number of such access attributes could be increased (see Haber's MCB-669). The proposed facility will permit general use of these extended access attributes, however, directory control could specifically use such added access attributes to more finely control access to directory attributes. For example, there could be the quota attribute that allows a process to modify the quota in the directory but not modify anything else in the directory. Of course, the application for which the extended access was developed was that of message segments which needs a set of access attributes other than the standard set. The argument against extended access is that such information can be kept in the segment itself and should not be cluttering the directory, the ideal contents of a branch being only those things which must be in the branch to allow the file system to function properly. The counterargument is that if the information is placed in the segment instead of the branch a separate mechanism almost identical to the file system access control mechanism would have to be created. The issue here is one of subsystem convenience versus system convenience.

An issue with a similar tradeoff is that of a segment type. It is necessary to be able to properly identify certain types of segments, e.g., message segments. Two methods of doing this have been proposed, a naming convention (Webber's memo of January 11), and a segment attribute (see my memo of December 31). The naming convention has the advantage of adding no new mechanism to the file system. It has the disadvantage of forcing restrictions on the use of names and having multiple names on segments. There are also problems when these segments are referenced through links whose names cannot be controlled. Of course, there is the third alternative of placing the segment type in the segment itself, but this has the problem of forcing the possible creation of a new system wide or at least ring wide convention at this late stage which would cause compatability problems.

The next issue has to do with the right to modify one of the most important segment attributes, the ACL. The confusion arises when rings are taken into account. The current mechanism is supposed to obey the rule that a process cannot place a ring number on an ACL that is below the current validation level. Also the user must have effective write permission in the segment as well as the directory containing the segment in the ring of the current validation level. This implies that a process cannot place a segment in a lower ring than that in which it is operating, but can remove a segment from a lower ring. Also it is possible for a process to manipulate access on a segment that may reside in a lower ring for another process if he does not modify that individual's access. A slightly different rule that could be used to revoke these privileges (note that I do not propose this change but only give it as an example of an alternative) is to not permit the modification if an ACL if any ring brackets currently on it are lower than the validation level. The whole issue of the modification of ACL's and its interaction with rings requires better definiton. The problem here is to decide what is permissible and what is not. Another associated proposal is that of making the ring brackets a segment attribute rather than a segment-user attribute. This would simplify the issue of who has access to modify the ACL of a segment since all users of the segment would have the same ring brackets, however, it is not clear if this is a useful or meaningful idea. To resolve this question we need to find meaningful examples of segments where different users require different ring brackets.

The last issue to be discussed here is complicated by all the above issues. This is the issue of the CACL. CACL's are supposed to be the logical extension of all the ACL's of the segments in the associated directory. This currently causes a problem because the access attributes of directories and non-directory branches do not have the same meanings and therefore entries on CACL's cannot meaningfully apply to both. Therefore, presently, ACL's can only be meaningfully applied to one of these two types of segments and individual ACL's must be used for the other type. This problem is worsened by the addition of extended attributes with an unlimited number of interpretations. Secondly, there is a problem with the right to modify CACL's. As stated above the current rule with respect to modifying ACL's is that the process must have effective write permission on the segment whose ACL is to be modified. Since modifying a CACL is equivalent to modifying every ACL, the process must have effective write access in every segment in the directory. Making this check is an expensive operation and will make modifying a CACL an expensive operation. Note that this problem is currently ignored leaving a hole in the current implementation. Again extended access complicates the problem because in the CACL of some segments permission to modify an ACL may be determined by the extended access attributes. One immediate reaction is to say that CACL's should be eliminated. However, CACL's perform an extremely useful service in that when a new segment is created in a directory it automatically has the access of the CACL without having to remember to add it each time. Assuming that we must preserve CACL's there is the possiblity that we can maintain logically multiple CACL's, i.e., a logical CACL for each segment type (assuming we can determine how to distinguish segment types). It would seem that any proper solution to the CACL problem will be expensive, a problem that should be overcome because, at least at present, CACL's are probably more heavily used than ACL's.

This concludes my sketchy discussion of the problems of segment attributes that have been brought to the surface by the introduction of the full ring mechanism. Many problems associated with rings have not been discussed like the problem of separating privileged subsytems in ring 3 as discussed in earlier memos by Clingen and Schroeder. These other topics will have to be considered later. The topics discussed here are those which should be resolved immediately, if not sooner, if they are not to cause bottlenecks, therefore, comments are requested as soon as possible.