PROJECT MAC

December 17, 1974

Computer Systems Research Division Request for Comments No.65

The Removal of the Use of Path Names From Ring 0

by Allen W. Luniewski

This RFC discusses one of the consequences of the proposed removal of name space management from ring 0 - the difficulty ring 0 routines will encounter when they try to use path names. Proposals are made to remove the use of path names from the normal operation of ring 0. Difficulties with the complete removal of the use of path names from ring 0 are pointed out.

A currrent area of research at Project MAC concerns the certification of a security kernel for Multics. The size and complexity of the current Multics supervisor makes certification difficult. To make the task of certification easier, investigations are underway to simplify the ring 0 resident supervisor, the most protected part of the supervisor. These simplifications involve the removal of mechanisms from ring 0 as well as the simplification of those remaining. Richard Bratt in RFC 49 presented a proposal to remove the name space management of a process from ring 0 to an outer ring. This removal will have the effect, among others, of removing the current ability of ring 0 to directly associate a segment number with the object named by a file system path name. Ring 0 currently has this ability by using the name space manager which resides in ring 0. This ability is important as many ring 0 procedures currently use path names in a way which requires them to obtain a pointer to the segment named by the path name. This pointer consists of essentially a segment number since it points to the base of the segment. The only mechanism which will be in ring 0 to associate a segment number with an object specified by a path name is by direct reference to the file system. In most cases such a reference is not really needed. An alternative method is available to obtain the segment number. In many cases, the path name that ring 0 has originally came as a parameter from an outer ring where the new name space manager will reside. It thus is reasonable for the outer ring to assume the responsibility of

obtaining a segment number corresponding to the path name by using the name space manager before calling the ring 0 procedure, and then passing the segment number, instead of the path name, as a parameter. There seems, on the surface at least, little need for ring 0 procedures to concern themselves with path names. The proposal outlines a way to remove the use of path names from ring 0.

In this report, we discuss work done on this proposal to remove ring 0's present use of path names. Work undertaken to date has been in two phases. In the first phase, an audit of ring 0 was conducted. This audit generated a list of modules in ring 0 which use path names. Also determined were how these modules obtain the path names they have and how they use those path names. The essence of these lists is contained in the appendices. The conclusion drawn from these lists was that the removal of the use of path names from ring 0 was not an impossible task or even a monumental one (in terms of the amount of work involved). Although the number of affected modules was large, no implementation problems were anticipated and the necessary changes appeared fairly simple. With this conclusion the second phase of work was begun - the modification of the ring 0 modules themselves. This has proceeded without difficulty and, as of this writing, is essentially complete. No major problems have been encountered. When completed, the result will be a new ring 0 which does not accept or use path names

explicitly. Subsequent paragraphs of this report will discuss the types of changes made, the problems encountered and the work remaining to be done.

When the audit of the ring 0 supervisor was completed, the changes necessary to remove the use of path names by ring 0 were found to fall into four categories. The first class of changes concerns parameters to procedures. Many modules currently accept a path name as one of their parameters. The change made is the obvious one. All such modules have been modified to accept segment numbers as parameters instead of path names. This means, of course, that callers of the changed module must be modified to pass a segment number instead of a path name to these modules. In particular, this impacts non-ring 0 procedures. This will be discussed later. The ring 0 modules have, however, been modified to use this new interface.

The second class of changes involves those modules which use text embedded path names (that is, a string constant such as ">process_dir_dir"). The solution is for such text embedded path names to be expanded by the module itself. By expanding the path name, it is meant that the module uses ring 0 primitives to generate a segment number corresponding to the path name. This is done by first obtaining the segment number of the root, then using it, by a call to initiate_, to obtain a segment number corresponding to the first component of the path name and then

using it... In this way we ultimately get a segment number corresponding to the segment named by the path name. As noted in appendix C text embedded path names occur infrequently and even then along infrequent control paths (system initialization, process creation and process destruction), thus the slight inefficiency of such expansion is unimportant.

The third class of changes concerns those modules which use the path name of the process directory, which is currently stored in pds$process_dir_name. For these modules, a new entry in pds is provided, pds$process_dir_segno, which contains the segment number of the process directory. This datum must be set when the process first wakes up in ring 0 (in the module init_proc) after being created.

The fourth and last class of changes occurs in system initialization. At the time of system initialization directory path names and entry names corresponding to segments are read from the MST (Multics System Tape) and saved in the SLT (Segment Loading Table). Later, during system initialization, these same names are used to add the corresponding segment to the file system. These path names may be arbitrary. In order to append these branches, we need a segment number corresponding to the directory named by the path name. There is no module in the ring 0 supervisor which will be able to obtain a segment number for this directory path name for this purpose. We are forced,

therefore, to have a module around at system initialization time to perform the parsing and initiating of directory path names. This parser represents the only ring 0 module which ever parses a path name. Into one of these four categories fall all of the ways that ring 0 obtains a path name.

The last paragraph would lead one to believe that there are no problems associated with removing the use of path names from ring 0. Unfortunately this is not the case. Problems occur in one specific and one general area. The specific area is the on_line_salvager which accepts a pointer to a directory segment as a parameter. Currently, the on_line_salvager then obtains a path name corresponding to this pointer. It uses this path name within its informative messages to the system programming staff. A simple replacement of a path name by a segment number (or pointer) will not do: a message containing a segment number (or pointer) is probably not very informative or very useful. The general class concerns serious error conditions. When some errors occur, the ring 0 module detecting the error attempts to report the error to the operator using the module syserr. The message passed to syserr frequently contains a path name. With path names no longer being passed around within ring 0, these modules will lose this ability, they will be forced to use a segment number in their messages. The resulting message will probably not be, as for the on_line_salvager, very informative or useful. In both of these cases the concern is with identifying

an object in the storage system within an informative message. Even with the removal of the name space manager from ring 0, it is still possible, with ring 0 data bases and primitives, to construct an identifying string for the segment from its segment number and then use this string in the message. Two forms for this string suggest themselves, a unique ID path name and a "standard" path name. The unique ID path name can be obtained from the KST (Known Segment Table) fairly easily. It consists of the unique ID of the superior directories of the segment as well as the unique ID of the segment itself. A "standard" path name can be constructed by looking within the superior directory of the segment for the entry of the segment. This gives one an ASCII representation of the branch name. Applying this to all the superior directories, we can construct an ASCII path name for use. Both alternatives are viable and not too hard to implement. Aside from these two areas, no major problems were encountered.

At this time not all work on this project is complete. Two areas need to be worked on. First, the items in the previous paragraph relating to the on_line_salvager and error conditions must be solved. Second, something must be done to cope with non-ring 0 routines which call a module with a changed interface. One clearly unacceptable solution is to require all such procedures to be changed to correspond to the new interface. The other solution is to insulate non-ring 0 routines from the changes by write arounds. Non-ring 0 calls to the changed

interface are transferred to an intermediate, non-ring 0, routine which can use the name space manager. This intermediate routine uses the name space manager to obtain a segment number corresponding to the object named by the path name it receives as a parameter. It then passes this segment number instead of the path name to the new ring 0 interface along with the other parameters. Thus non-ring 0 procedures need not know of the new interface, they will work just as if nothing had changed. New non-ring 0 procedures should be aware of the new interface and use it when possible. With the completion of these items, Multics will be ready to operate without using path names within ring 0.

In this paper a review of work done to remove the use of path names within ring 0 has been presented. The significant conclusion to be drawn is one of feasibility. It has been found to be quite practical to remove the use of path names from ring 0. The necessary changes, while numerous, present little conceptual difficulty. Instead they represent fairly minor textual changes. The resulting changes to ring 0 are, however, not significant in their own right. The hope is that a review of the remaining items, in particular the file system, in ring 0, given the removal of name space management and path names from ring 0, will reveal mechanisms and interfaces which are not needed in the security kernel. Such a review does have the potential for significant simplifications to the kernel.

Appendix A

A List of Affected Modules

This appendix lists all modules known to be affected by the proposal presented in this report. All affected modules are segments of the ring 0 supervisor or are system initialization routines. Subsequent appendices break down this list into groups of modules with similar changes.

acl

act_proc

append

asd_

assign_device

bound_fault_handler

chname

deact_proc

delentry

del_dir_tree

ex_acl

force_access

fs_move

fs_search

gim_assign

initiate_search_rules

init_branches

Appendix A

A List of Affected Modules

init_proc

level_0_

list_dir

load_system

makecls

makestack

make_branches

make_seg

move_device

ncp_main_

on_line_salvager

quota

rest_of_datmk_

ringbr_

set

star_

status

status_

truncate

Appendix B

Parameters

These modules currently take a path name as a parameter, they are modified to take a segment number instead. When only a module is listed, all entry points of that module are affected except those listed in Appendix C.

acl

act_proc

append

asd_

chname

delentry

del_dir_tree

ex_acl

level_0_

list_dir

make_seg

move_device

quota

ringbr_

set

star_

status

status_

truncate

Appendix C

Entry Points Not Affected by the Changes in Appendix B

asd_$set_exmode_level

delentry$dsegno

quota$check

quota$cused

set$bc_auth_ptr

set$bc_seg

set$max_length_ptr

set$safety_sw_ptr

status_$get_max_length_ptr

status_$get_safety_sw_ptr

status_$mins

Appendix D

Modules Internally Expanding a Path Name

Module	Path Name Expanded
act_proc	">pdd" and ">pdd" || some_unique_name
deact_proc	">pdd"
init_branches	">" and ">pdd" and ">system_library_1"

Appendix E

Miscellaneous Changes

append$branchr

Parameter three is deleted (it refers to links).

init_branches$branch

Returns in a new sixth parameter the segment number corresponding to its first parameter.

makestack

Refers to pds$process_dir_segno for the segment number of the process directory.

make_branches

Returns in a new eighth parameter the segment number corresponding to its first parameter. Also it calls a new module, initiate_path_name, to expand a path name to a segment number.

make_seg

Parameter three is deleted (it is a reference name). Also it refers to pds$process_dir_segno for the segment number of the process directory.

Appendix F

Affected Modules Not Yet Modified

bound_fault_handler

force_access

fs_move

fs_search

gim_assign

initiate_search_rules

init_proc

makecls

ncp_main_

on_line_salvager

rest_of_datmk_