file in MSPM

6-1-70 DRAFT

date

Identification

validate_arg
R. K. Kanodia

Purpose

When an inner-ring procedure is called by an outer-ring procedure, the called procedure has, by definition, greater access privileges than does the calling procedure. Some means must be furnished within the protection mechanism to enable the inner-ring procedure to determine that the arguments passed to it will not cause it to exercise those privileges unwisely. Procedure validate_arg, discussed in this section, performs such argument-checking.

Introduction

When an attempt is being made to transfer control from one ring to another, a ring-crossing fault occurs. The Fault-Interceptor, then, calls a special ring 0 module, called the Gatekeeper, to exercise control over all inter-ring calls and returns. The Gatekeeper performs several tasks in handling this transfer of control between rings.

When an inner-ring procedure is called by an outer-ring procedure, the outer-ring procedure will, in general, supply an argument list (a list of pointers to actual arguments or their specifiers) with the intention that the target procedure use some of these locations (pointed to by the argument pointers) to read information and some to store information on its behalf. Inner-ring procedures, by definition, have greater access-privileges than the outer-ring procedures. A calling procedure could then, deliberately or otherwise, supply argument pointers to a segment

space to which it does not have access(but to which the procedure in the target ring does have access). To prevent this from happening, an effort must be made to ensure that the outer-ring procedure does indeed have the necessary access rights to all the segments specified by the pointers in the argument list. The Gatekeeper calls on validate-arg to perform this task.

On inward calls there is another type of protection violation that must be prevented. It concerns the possibility of deliberate or accidental changes to the argument pointers after they have been validated, but before they have been used by the inner-ring procedure. To prevent this postvalidation tampering, every argument pointer is first copied into an inner-ring stack segment and then validated from there. (The copied argument list and specifiers in the inner ring stack segment are then passed to the inner-ring procedure rather than the original argument list.)

Method

validate_arg has two inputs:

1) the calling procedure's argument list
2) validation data from the gate (i.e. specification of the arguments the called procedure expected)

The validation data specifies the number of arguments expected and for each argument (a) the type of argument and (b) whether the argument is input or return. (This information is used to determine the access necessary for each argument).

validate_arg first ensures that the caller has access to the segment in which the argument list exists. It then compares the number of arguments that the argument list has against the number of arguments specified in gate.

After ensuring that both numbers are the same, the main loop, which in each cycle "validates" and copies one argument, begins. Specification in validation data specify what type of argument is expected. If the argument is a non-string-scalar it copies the argument pointer into the new argument list and then checks access on the segment that contains the data. If it is an input argument the "read" is required, for a return argument "write" is required.

For strings in an EPL argument list, the argument pointer points to a specifier. The Specifier then has pointers for data, dope and free storage areas. Appropriate access checks are made on each of these segments and modified argument pointer and specifiers are copied into the area provided for the new argument list.

There is an important and special case of arguments that are themselves pointers. Pointers are in general supplied by calling procedure, with the intention that the target procedure use the argument as an indirect reference to a data area. The target procedure then either reads from this area or writes into this area. This type of argument is called pointer-to-data type. For this type of argument not only that modified argument pointer is placed into new argument list, the value of the argument (which itself is a

pointer). ~~is also~~ copied. The "input" or "return" type depends on whether the target procedure reads from the data area or writes into it, even though the argument itself is supplied by called procedure.

In case of PL/1 argument list if argument descriptors are present they are also checked for access and copied.

For "checking access" validate_arg examines the segment-descriptor word in the descriptor-segment of calling procedure's ring; if the segment is not known to the process, it calls on the procedure fs_get$mode.