MULTICS TECHNICAL BULLETIN

TO: Distribution

FROM: Melanie B. Weaver

SUBJECT: Runtime Parameter Checking

DATE: July 11, 1974

INTRODUCTION

A fair number of bugs by Multics users are caused by failure to pass the correct number or types of arguments expected by the callee. This number would be greatly reduced if the system were to check that the arguments matched the target's intended parameters. To accomplish this, it is proposed that compilers associate parameter descriptors with links and that these be compared with the target entry's parameter descriptors by the linker and the binder. Since the actual target found during linking may differ each time a given link is snapped, it is possible for an otherwise debugged procedure to mismatch its calls to an external entry; therefore the default will be to check all links. A command/subroutine will be provided to turn off the checking.

This MTB presents the proposed link changes and describes the modifications to system modules necessary to implement the checking. In order to simplify the following discussion, we shall refer to the descriptors generated from entry and procedure statements, i.e. incoming, as _parameter descriptors_ and to the descriptors generated for links from the entry declarations, i.e. outgoing, as _argument descriptors_. Note that the actual descriptors passed with an arument list will not be used, as they will not be available at the proposed time of checking; the linker does not have a pointer to the argument list and there is no argument list at the time of binding. Because the descriptors will be created from declarations only, they must match exactly, according to language rules.

Parameter descriptors are already being produced by PL/I and fortran and are associated with an entry's definition. The complete, though out of date, definition is in section 11.3 fo the SWG. Briefly, a definition contains flags which indicate whether a parameter count and/or descriptor offsets are appended to the definition. Parameter descriptors themselves live in the text section.

A PROPOSED NEW LINK TYPE

We will now discuss the proposed link changes. Above all, we would like to avoid an incompatible object segment change that would force the linker and binder to determine whether they had the new object segment format before they could check. The simplest solution seems to be to have a new type of link, type 7, which would be analogous to type 4 (segname|entryname) except that the high order half of the trap word would point to a descriptor offset array. Currently only type 4 links are used when calling entrypoints; if and when type 2 (indirect through pointer register) links are used, still another analogous type can be defined. This of course leads to the restriction that no type 7 link can have a trap before link, but that does not affect PL/I and fortran and should not inconvenience other languages. An advantage of this solution is that it allows one to distinguish between links with zero arguments, which should be checked, and links with an indeterminate number of arguments, i.e. declared options(variable), which should not be checked. The latter will remain type 4 links. The descriptors themselves will be in the text section as unique constants the way other descriptors are now. In many cases they will have been used in setting up the argument lists. The proposed declaration for the structure pointed to by the trap word is as follows:

```
declare    1 link_parameters based aligned,
             2 n_args fixed bin,
             2 array (* refer(n_args)) aligned,
               3 nwords bit(18) unaligned,
               3 offset bit(18) unaligned;
```

1) n_args   is the number of arguments to be passed.

2) nwords   is the number of words occupied by the descriptor.

3) offset   is the offset of the descriptor in the text section.

The nwords field exists because array and structure descriptors are several words long. It enables two descriptors to be compared without having to interpret the descriptors themselves to determine lengths.

LINKER CHECKING

Now we will describe the general checks to be made and how they will affect the user interface. Comparisons can be made only if there are both outgoing and incoming descriptors; thus no checks will be made when there is not a type 7 link or parameter descriptors for the entry. When a type 7 link is encountered, the two parameter counts are compared, if they exist. If that succeeds, and the entry has valid parameter descriptors, the descriptors are compared one by one with a character string template. If the parameter count is the only valid information, the checking will stop after that.

The linker will, in addition, use the switch pds$check_descriptors to determine whether or not it should check. It is proposed that the default for this switch be on, keeping in mind that it affects only procedures compiled with the new link type and thus will have no immediate effect. In any case, there will be commands to turn the switch on and off: parameter_check_on (pcn) and parameter_check_off (pcf). There will also be the command skip_parameter_check (spc) to disable checking only for the next link that would otherwise be checked. This will utilize the switch pds$temp_check_descriptors, which will be off only between the time the command is invoked and the time the next link to be checked is encountered. If any part of the parameter check fails, the linker will signal linkage_error with the new code error_table_$mismatched_params ("Supplied parameters do not match target parameters."). If the user wishes, he can type "parameter_check_off" or "skip_parameter_check" and then "start" to continue. The system's default condition handler will not say which parameter was in error.

PERFORMANCE ESTIMATE

Adding these checks will degrade performance, but only slightly. A rough calculation was made to determine the extra percentage of CPU time that would be taken up. A version of link_snap with the checks was coded and compiled with the EIS compiler. Assuming a worst case where all the metered links (types 3, 4 and 6) become type 7 links with 3 arguments, the added code is about 25% of the current link_snap code. According to a recent run of system_link_meters, link_snap itself takes less than 10% of total linking time, and the linker takes approximately 1 % of total CPU time (usually much less). Thus the checks would add at most about .025% to CPU time spent linking. This percentage may increase slightly when the rest of the hardcore segments involved are converted to EIS. This estimate does not include any extra page faults taken on the descriptors themselves. The descriptor pointer arrays are in the definition sections and probably on the same pages as the link and definition information used by the linker anyway. The argument descriptors may have been referenced when the argument list was set up just prior to invoking the linker and so may still be in core, depending on how much room the segment search, linkage section management, etc. took. The parameter descriptors will be referenced by the target's entry sequence within a few instructions and so need to be in core anyway. Thus it does not appear that parameter checking will be too costly.

BINDER CHECKING

The binder will also check parameters by default; however it will never consider a mismatch to be a fatal error. The function of the checking will merely be to print any resulting error messages. It is proposed to check only on a global scale, with a -no_check (-nck) control argument to inhibit it; thus there is no need for any bindfile changes. When checking, there are two types of situations that would produce messages. One is when the binder is prelinking to another component and the descriptors or number of parameters do not match. The messages in this case will be:

    Incorrect argument types passed from caller to callee.
    Wrong number of arguments passed from caller to callee.

The other situation is when the binder is regenerating external links and encounters two or more links to the same target but with different parameter descriptors. This difference is not enough to justify generating separate links, although no one set of parameter descriptors can be used for all references, so the message will be:

    Multiple links to target have differently declared
    argument lists.
    No parameter descriptors will be used.

Only one such message will be printed per target. These messages will result solely from comparisons of type 7 links. If all type 7 links to a target match, but there are also some type 4 links to the same target, the common link will be type 4 and no message will be printed for that case.

OTHER SYSTEM COMPONENTS AFFECTED

Introducing the new link type will affect other parts of the system as well. The fortran and PL/I compilers must be changed to produce it. basic and alm will not be affected at present; basic does its own nonstandard argument checking and alm has no syntax for specifying parameter types, either incoming or outgoing. This means that links to gates, which are in alm, will not be checked. Hopefully sometime in the future appropriate pseudo ops can be defined for alm enabling descriptors to be generated for entries and links. Both the prelinking part of system initialization and the MST checker must know about the new link type. It would be useful for the checker to compare descriptors also, but this is not planned for the near future.

If gates are ever checked, there is a potential problem that will arise if the linker is moved to the user ring in that the parameter descriptors are stored in the text section. No solution is being proposed here.