Subject: MCR-10025, Repair Code Broken by command_processor_ Change

Author: Gary Dixon

Date: January 9, 2017

In 1986, MCR7375 introduced a change to the arg_list structure, based upon a false assumption. This induced several bugs in existing code.

In this MCR-10025, these bugs are identified and repairs for them are proposed.

The discussion begins with a brief introduction to the earlier MCR, and its false assumption.

MCR7375

MCR7375 introduced a change to the command_processor_. This routine constructs an argument list used to pass command-line arguments to a command entrypoint. The passed arguments are command options (parameters for the command entrypoint) typed by the user in the command line. The same mechanism is also used within a subsystem to construct the argument list for each subsystem request from a request line typed by the user.

The changes in MCR7375 included the following:

- Add an incompatible variant of the arg_list structure, to append a command (or request) name ptr and length to the end of the structure. To use this variant structure, a new flag defined in a pad field of the arg_list structure is set true. See further details in the discussion below.
- Add two new entrypoints to obtain the command/request name from an arg_list if it contains the name information. cu_$get_command_name_rel and cu_$get_command_name entry points were added to cu_.alm.
- Change command_processor_.pl1 to construct arg_lists of the command name variant.

MCR7375 Changes to arg_list Structure

The original arg_list structure is shown below, followed by a variant used when an environment_ptr is passed as part of the argument list. The environment_ptr is passed for calls to internal procedures, and calls through entry variables. This variant is used if arg_list.call_type is set to Envptr_supplied_call_type. Otherwise, the arg_list variant is used.

```
dcl 1 arg_list          aligned based,
      2 header,
        3 arg_count     fixed bin (17) unsigned unal,
        3 pad1          bit (1) unal,
        3 call_type     fixed bin (18) unsigned unal,
        3 desc_count    fixed bin (17) unsigned unal,
        3 pad2          bit (19) unal,
      2 arg_ptrs        (arg_list_arg_count refer (arg_list.arg_count)) ptr,
      2 desc_ptrs       (arg_list_arg_count refer (arg_list.arg_count)) ptr;

dcl 1 arg_list_with_envptr aligned based,  /* used with non-quick int and
                                              entry-var calls */
      2 header,
        3 arg_count     fixed bin (17) unsigned unal,
        3 pad1          bit (1) unal,
        3 call_type     fixed bin (18) unsigned unal,
        3 desc_count    fixed bin (17) unsigned unal,
        3 pad2          bit (19) unal,
      2 arg_ptrs        (arg_list_arg_count
                             refer (arg_list_with_envptr.arg_count)) ptr,
      2 envptr          ptr,
      2 desc_ptrs       (arg_list_arg_count
                             refer (arg_list_with_envptr.arg_count)) ptr;

dcl (
     Quick_call_type         init (0),
     Interseg_call_type      init (4),
     Envptr_supplied_call_type
                             init (8)
     )         fixed bin (18) unsigned unal int static options (constant);
```

Neither form of arg_list includes a version number. So changes to the structure are difficult to make in a compatible fashion. Furthermore, this structure is an artifact passed as part of the Multics Calling Sequence (see *Subroutine Calling Sequences* in the MPM Subsystem Writer's Guide, Section 2 of AK92-02). This means the arg_list structure is referenced directly by various programming language runtime code (e.g., pl1_operators_, pascal_operators_, fortran_operators_); by inline code sequences generated by language compilers; and by other low-level utilities (e.g., command utility (cu_) subroutines).

The change introduced by MCR7375 appends command or request name information to the end of the arg_list structure variant, if a new flag is set to true. The new information was appended only to the arg_list variant, because that is the only form passed to entry points invocable by the command_processor_ (external commands, and subsystem requests).

When using the variant, a .has_command_name flag is inserted in the middle of the arg_list.pad2. Seemingly an odd location for the new flag, it was intentionally positioned as bit 18 of the word containing the arg_list.desc_count and .pad2 elements; bit 18 is the first bit of the lower half of this word.

MCR7375 assumes that all code referencing desc_count accesses that value using halfword loads from the upper halfword; the lower halfword is ignored. Recently, some code has been found that does not adhere to this assumption.

The arg_list variant including the new has_command_name flag, and the trailing name substructure is shown below.

```
dcl 1 command_name_arglist       aligned based,
      2 header,
        3 arg_count              fixed bin (17) unsigned unal,
        3 pad1                   bit (1) unal,
        3 call_type              fixed bin (18) unsigned unal,
        3 desc_count             fixed bin (17) unsigned unal,
        3 mbz                    bit(1) unal,
        3 has_command_name       bit(1) unal,
        3 pad2                   bit (17) unal,
      2 arg_ptrs                 (arg_list_arg_count
                                  refer (command_name_arglist.arg_count)) ptr,
      2 desc_ptrs                (arg_list_arg_count
                                  refer (command_name_arglist.arg_count)) ptr,
      2 name,
        3 command_name_ptr       pointer,
        3 command_name_length    fixed bin (21);
```

Most code does follow the MCR7375 assumption. The desc_count value is accessed by loading the upper halfword into an index register. Code for cu_$arg_ptr shows how this access is typically performed:

```
arg_ptr:  eppbp     sp|stack_frame.arg_ptr,*      " ptr to caller's arg_list

arg_ptr_common:
          ldaq      null                          " initialize output values
          staq      ap|4,*                        " ... null argument pointer
          stz       ap|6,*                        " ... zero length
          stz       ap|8,*                        " ... zero error code

          lda       ap|2,*                        " pick up argument number
          tmoz      arg_ptr_noarg                 " ... must be positive
          als       1
          eax1      0,al                          " 2*argument_idx -> X1
          cmpx1     bp|0                          " check against arg count
          tpnz      arg_ptr_noarg                 " ... arg_idx is too large

          eppbb     bp|0,1*                       " copy the argument pointer
          spribb    ap|4,*

          ldx2      bp|1                          " get descriptor word count
          tze       arg_ptr_no_descriptors        " ... no descriptors
```

It is clear from this code that the pad fields in all arg_list structure variants are not optional; all pad fields must be set to “0”b for the above code to work correctly. These padN fields probably should have been named mbzN, with a comment stating they must be set to zero.

Several important subroutines do not access the desc_count as above, and therefore do not adhere to the assumptions made in MCR7375. These subroutines now have bugs induced by the MCR7375 change.

MCR-10025 Bug Repairs

Change cu_.alm: cu_$arg_ptr and cu_$arg_ptr_rel share common code that returns ptr and length of the Nth argument string, given input argno N. This code checks whether arg_list.desc_count=0, in order to avoid accessing the argument's descriptor string length if no descriptors are given. Current code loads the entire word containing desc_count and pad2 into the A register. If zero, then no descriptors are present.

Users of the new command_name_arglist variant may sometimes not create descriptors (.desc_count = 0), even if arguments not requiring descriptors are supplied. The command_processor_ can supply a 0 desc_count when no arguments appear in the command line. Since the new .has_command_name flag is set in the lower halfword, the current code would fail to detect lack of descriptors.

The proposed change to cu_$arg_ptr and cu_$arg_ptr_rel (common code) is shown below.

```
arg_ptr:  eppbp     sp|stack_frame.arg_ptr,*      " get pointer to caller's argument list

arg_ptr_common:
          ldaq      null                          " initialize output values
          staq      ap|4,*                        " ... null argument pointer
          stz       ap|6,*                        " ... zero length
          stz       ap|8,*                        " ... zero error code

          lda       ap|2,*                        " pick up argument number
          tmoz      arg_ptr_noarg                 " ... must be positive
          als       1
          eax1      0,al                          " 2*argument_idx -> X1
          cmpx1     bp|0                          " check against the argument count
          tpnz      arg_ptr_noarg                 " ... argument_idx is too large

          eppbb     bp|0,1*                       " copy the argument pointer
          spribb    ap|4,*

          lda       bp|1                          " get descriptor word count
Changed to:
          ldx2      bp|1                          " get descriptor word count

          tze       arg_ptr_no_descriptors        " ... no descriptors
          adx1      bp|0                          " compute offset to the descriptor
          lxl0      bp|0                          " ... check for an environment pointer
          anx0      8+2,du
          tze       2,ic
          eax1      2,1                           " ... skip over environment pointer

          lda       bp|0,1*                       " pick up the descriptor
          tmi       *+2
          ana       =o777777,dl                   " mask for version 1 descriptors
          ana       descriptor_mask               " mask for version 2 descriptors
          sta       ap|6,*                        " return the argument length

arg_ptr_no_descriptors:
          short_return

arg_ptr_noarg:                                    " unknown argument specified
          lda       error_table_$noarg
          sta       ap|8,*
          short_return
```

The ldx2 instruction loads only the upper halfword (containing .desc_count and .mbz) into the index register, and sets the indicators based upon that halfword value. The tze instruction that follows therefore ignores any .has_command_name flag set in the structure.

Change pascal_operators_.alm: the return_zero operator examines the last argument passed to an external Pascal program entrypoint. If it is a fixed bin(35) aligned number, this operator sets the argument to zero (as an entrypoint return status).

This operator code violates the MCR7375 assumption, by attempting to test for arg_list.desc_count=0 directly in memory, using a sze instruction that sets indicators based upon contents of the entire memory word. This code fails to detect desc_count=0 if the has_command_name flag is set in the lower halfword.

The proposed repair for pascal_operators_.alm is shown below.

```
A2973                   epp1        6|stack_frame.arg_ptr,*
A2974                   szn         1|1
A2975                   tze         3|0
A2976                   ldaq        1|0
A2977                   eax1        0,qu
A2978                   qls         1
A2979                   ana         8+2,dl
A2980                   tze         +2,ic
A2981                   adq         2,du
A2982                   lda         1|0,qu*
A2983                   ana         neg_mask
A2984                   cmpa        fb35_desc
A2985                   tnz         3|0
A2986                   stz         1|0,x1*
A2987                   tra         3|0
Changed by B to:
A2973                   epp1        6|stack_frame.arg_ptr,*
B2974                   ldx1        1|1
B2975                   tze         3|0
B2976                   eaq         0,x1
B2977                   qls         1
B2978                   lda         1|0
B2979                   eax1        0,au
B2980                   ana         8+2,dl
B2981                   tze         +2,ic
B2982                   adq         2,du
B2983                   lda         1|0,qu*
B2984                   ana         neg_mask
B2985                   cmpa        fb35_desc
B2986                   tnz         3|0
B2987                   stz         1|0,x1*
B2988                   tra         3|0
```

The sze instruction is replaced by ldx1, to examine only the upper halfword containing the desc_count*2. That count is then moved into the Q upper register, and multiplied by 2.

The arg_count*2 and call_type are loaded into A register. Then the arg_count*2 is copied to x1, as offset to pointer to the last argument's storage.

Finally, if an envptr is present, include it in the Q upper offset by adding 2,dl to Q register. Then the final descriptor is loaded into the A register, using the offset now in Q upper.

This change allows a pascal program having no arguments to be called as a command, without encountering a fault in referencing an arg descriptor when no arguments/descriptors are supplied in the argument list.

Other code in pl1_operators_.alm, pascal_operators_.alm and cu_.alm was examined for failure to match the assumption of MCR7375. Only the two cases above were found.

Version History

Date	Revision	Author	Comment
2017-01-07	0.1	Gary Dixon	Initial version.
2017-01-07	0.2	Gary Dixon	Redo changes to more closely fit original code.
2017-01-08	1.0	Gary Dixon	Make text describing pascal operator change agree with final version of new code.