Subject: MCR10109, Minor fixes for mbuild subsystem and bind command

Author: Gary Dixon

Date: December 25, 2021

Ticket: https://multics-trac.swenson.org/ticket/270

Issue #1

As described in Ticket 270, mbuild issues an incomplete and confusing warning if the user gives the wrong library name for a segment being replaced or updated. For example, with an installation directory containing one file to be replaced, and the following build script:

```
list

Segments = 2, Lengths = 4.

r w    3  probe_info_requests_.pl1
r w    1  test_mb_01.mb

r 14:43 0.083 0


print *.mb

Installation_directory:  >udd>m>gd>w>test_mb_01;

 Build_script:  test_mb_01.mb;

 Bound_obj:     bound_probe_                        IN: tools  UPDATE;
   source_arch: bound_probe_.2.s.archive             UPDATE;
     source:    probe_info_requests_.pl1            REPLACE        compiler: pl1 -ot;
r 14:43 0.087 0
```

Because bound_probe is actually installed in the sss library, mbuild's read request issues a warning message. But that messages does not adequately tell the user that the given library name is incorrect.

```
mbuild

Installation_directory: >udd>m>gd>w>test_mb_01

           Scanning, with read of any build_script...


mbuild (scan): Do you want to read the test_mb_01.mb file?   y

Warning 17:  Seg found in installation directory does not match statement on LINE 8
                   found Seg.name: probe_info_requests_.pl1          IN: sss.source
SOURCE:       source:              probe_info_requests_.pl1           REPLACE;

mbuild:  analyze
…
```

If the user does not change the library in the build script and read in the revised file, mbuild assumes the change is ADDing a new bound segment to the tools library.

```
mbuild:  print

 Build_script:              test_mb_01.mb;

 Bound_obj:                 bound_probe_                        IN: tools  ADD;
   source_arch:             bound_probe_.2.s.archive             ADD;
     source:                probe_info_requests_.pl1             ADD;
```

The proposed correction for this issue makes the read request display a fatal error about the bound_probe_ object segment identified 2 statements earlier in the build script, which says bound_probe_ is to be UPDATEd in the tools library. The fatal error 18 says bound_probe_ does not exist in the tools library, so therefore cannot be updated.

Then the original warning 17 clarifies by indicating that the probe component being changed was found in the sss.source library directory.

These two messages give the user all the information needed to correct the build script. With message 18 being a fatal error, the read operation terminates unsuccessfully preventing the build from proceeding until the problem is corrected.

```
mbuild

Installation_directory: >udd>m>gd>w>test_mb_01

           Scanning, with read of any build_script...


mbuild (scan): Do you want to read the test_mb_01.mb file?   y

FATAL ERROR 18:  Seg being UPDATEd does not exist in library given on LINE 6
SOURCE:   Bound_obj:               bound_probe_              IN: tools  UPDATE;

Warning 17:  Seg found in installation directory does not match statement on LINE 8
                  found Seg.name: probe_info_requests_.pl1  IN: sss.source
SOURCE:       source:              probe_info_requests_.pl1   REPLACE;

mbuild (read): Improper data in build script file.
                Parsing build script: >udd>m>gd>w>test_mb_01>test_mb_01.mb
```

Issue #2

While testing this change, an unusual problem occurred when the mbuild archive_prep request invoked the bind command to bind bound_probe_. The error is shown below in red text.

```
mbuild:  archive_prep

---------- library_fetch bound_probe_.2.s.archive -lb sss.source

   archive u bound_probe_.2.s.archive probe_info_requests_.pl1


---------- library_fetch bound_probe_.2.archive -lb sss.object

   archive udf bound_probe_.2.archive probe_info_requests_


---------- Sorting archives that were modified.

   archive_sort bound_probe_.2.s.archive

   archive_sort bound_probe_.2.archive


----------  bind  >ldd>sss>object>bound_probe_.1.archive bound_probe_.2.archive

bind: No write permission on entry. >ldd>sss>object>bound_probe_.1.archive
mbuild (archive_prep): Bind did not produce a Bound_obj segment. BOUNDOBJ:
bound_probe_
```

Why would the binder need write access to an object archive containing components to be bound into a bound segment? ANSWER: It doesn't need such access.

Investigating bind.pl1, the following code line initiates the bound object archive segments using an access_mode argument to specify the required access:

```
print bind.pl1 -fm 340 -for 2

          call initiate_file_ (archive_dname, archive_ename, access_mode,
               inp.archive (archive_idx).ptr, inp.archive (archive_idx).bc, code);
```

But the access_mode variable is never set before making this call, so stack garbage gets passed as the minimum required access mode.

It is unclear why this has not caused problems is past uses of bind invoked from command level. When called by mbuild, bind is an ssu_ request defined by the mbuild request table to invoke the Multics bind command. The stack garbage at the storage location for the unset access_mode variable contained the value "001"b or W_ACCESS (the constant from access_mode_values.incl.pl1). So initiate_file_ returned error_table_$no_w_permission which caused the error message generated by the bind request.

To correct this problem, bind.pl1 will be changed to assign R_ACCESS to the access_mode variable just prior to calling initiate_file_.

Issue #3

Another obscure message comes from mbuild's scan request when the install directory includes an exec_com segment.

```
mbuild

Installation_directory: >udd>m>gd>w>MCR10108

           Scanning, with read of any build_script...

mbuild (scan): Unexpected event. Seg(exec_com): setup_sysadmin_admin.ec
               differs from seg_name found by path_components: acct_start_up.ec
  >tools>acct_start_up.ec
```

This message occurs because mbuild_scan_.pl1 wants to reference most segments by their longest name, which is usually the name most significant to users. However, locating a segment in the Multics Libraries references each library segment by its first name (assuming that past installations have identified each segment by the name most significant to users, and making that the primary name on the segment.

Code will be added to mbuild_scan_ to always reference each exec_com segment (XXX.ec) by the first name on that segment as it appears in the installation directory. This matches the expectations when trying to locate installed versions of exec_com files in the Multics libraries.

Proposed Changes

The following build script shows the source programs being changed:

```
Description:
 A) Make mbuild's read request diagnose a build script line that references
    an existing segment using the wrong library name.
 B) Fix minor problem in bind command not setting required access mode
    when initiating object archives/segments being bound together.  R_ACCESS
    will be the minimum required access to such segments.
 C) Fix minor problem in installing exec_com segments: always use their
    primary name when locating them in the library.

Installation_directory:  >udd>m>gd>w>mb;

 Build_script:        mb.mb;

 Bound_obj:           bound_binder_           IN: sss  UPDATE;
     source:          bind.pl1                 REPLACE   compiler: pl1 -ot;

 Bound_obj:           bound_mbuild_                     IN: tools  UPDATE;
     source:          mbuild_scan_.pl1         REPLACE   compiler: pl1 -ot;
     source:          mbuild_script_parse_.rd  REPLACE   compiler: rdc -ot -trace off;
```

Issue #1: mbuild_script_parse_.rd compare_ascii output

The change adds a fatal (F-flag) message noting that a segment marked for REPLACE or UPDATE was not found in the library given in the build script line. mbuild_library_$locate_Seg is called to search the library source directory for the segment's name. No search if performed if the build script line does not specify a library name.

```
A534         "17 W S +: Seg found in installation directory does not match statement^/^s ^2-^a")
Changed by B to:
B537         "17 W S +: Seg found in installation directory does not match statement^/^s ^2-^a"),
B538        nonmatch_library init (                          /*  Err_char()               */
B539         "18 F S  : Seg being ^ad does not exist in library given")


Inserted in B:
B682        dcl  mbuild_library_$locate_Seg entry (ptr, ptr);
Preceding:
A677        dcl  mbuild_library_$replace_library_component entry (char(*) var, char(*) var,
char(*) var) returns(char(32) var);


Inserted in B:
B793
Preceding:
A787           if  depthI = 0  then return;    /* Do nothing if not parsing a major/minor
statement.     */


A792    if  tier(1).operation ^= "DELETE"  then      /*  operation values, for use in their child
tiers.       */
A793         return;                                 /* DELETE of these types is the exception.
Return a      */
A794       end;                                      /*  Seg() structure for any Bound_obj or
Unbound_obj that */
A795    end;                                         /*  is being DELETEd.
*/
Changed by B to:
B799                                                 /*  operation values, for use in their child
tiers.       */
```

```
B800    if  tier(1).operation  = "ADD" then          /* ADD means object is not yet in libraries.
*/
B801        return;
B802    else if  tier(1).operation = "DELETE" then;  /* DELETE of these types is the exception.
Return a      */
B803                                                 /*  Seg() structure for any Bound_obj or
Unbound_obj that */
B804                                                 /*  is being DELETEd.  This is done below.
*/
B805    else if  tier(1).library ^= ""  then do;     /* UPDATE and REPLACE requires only
verifying that the    */
B806                                                 /*  Seg() actually exists in any library
specified.       */
B807         level = tier(1);
B808         call mbuild_library_$locate_Seg( addr(bld), addr(level) );
B809         if  level.operation = "ADD"  then
B810              call Err_char( nonmatch_library, tier(1).operation );
B811              return;
B812              end;
B813         else return;
B814      end;
B815   end;
```

Issue #2: bind.pl1 compare_ascii output

The bind command change includes the access_mode_values include file which defines named constants for the various access mode combinations. It sets the required access_mode to the R_ACCESS needed for input archives/segments to the bind operation.

```
Inserted in B:
B339                    access_mode = R_ACCESS;
Preceding:
A331                    call initiate_file_ (archive_dname, archive_ename, access_mode,


A598
Changed by B to:
B607      %page;    %include access_mode_values;
```

Issue #3: mbuild_scan_.pl1 compare_ascii output

The change to mbuild scan request adds a variable to hold the suffix used for exec_com files; adds a RULE comment indicating that primary name on exec_com files is always used to identify the Seg(exec_com) data structure; and adds code to implement that new rule.

There is evidence that the algorithm choosing the longest name on a segment as its identifier does not work as well as anticipated. There are already 4 different exception types to this rule. If another exception is found, perhaps that algorithm should be scrapped, reverting to always using first name on the segment as it appears in the installation directory. But that decision can be delayed to another day.

```
A130        dcl  suffix_Info char(12) var;
A131        dcl  suffix_Build_script char(12) var;
Changed by B to:
B133        dcl (suffix_exec_com,
B134             suffix_Info,
B135             suffix_Build_script) char(12) var;
```

```
Inserted in B:
B141           suffix_exec_com     =
mbuild_info_find_$suffix_for_build_type("exec_com");     /* = ".ec"                */
Preceding:
A137
A138      STAR_BRANCHES:                                 /* Walk array of branches
returned by hcs_$star_          */


Inserted in B:
B152  /* RULE: Seg().name: always use longest entryname; except use first name
for .ec, its how ID'd in library.  */
Preceding:
A147  /* RULE: Seg().name: always use longest entryname; except use first name if it
begins bound_                */


Inserted in B:
B157          else if index (reverse (rtrim (star_names(nmFirstI))), reverse
(suffix_exec_com)) = 1 then
B158               long_name = rtrim (star_names(nmFirstI));
Preceding:
A151          else if index (star_names(nmFirstI), prefix_Bound_obj) = 1  then
```

Testing

The mbuild changes were tested by creating an installation directory which reproduced the original symptom of the issue using the installed mbuild; and then corrected the issue by using the changed version of mbuild and bind command. All three changes were used in tests of each issue. In the case of issue #1, the bogus library name was then corrected in the build script, and the corrected mbuild was used to build files in the installation.

Finally, the revised mbuild was used to prepare the installation directory for this MCR, to ensure that no new mbuild issues were introduced by the proposed changes.

All testing completed with successful outcomes in avoiding the original issues and introducing no new issues.

Documentation

No user interfaces are changed by this MCR proposal, so no documentation will be updated or added.

Version History

Date	Revision	Author	Comment
2021-12-25	1.0	Gary Dixon	Original version of the MCR proposal.