Subject: MCR10079, mbuild Enhancements and Minor Repairs

Author: Gary Dixon

Date: 2020-01-30

Multics ticket #196 suggests an enhancement to the new mbuild command to further guide the developer in performing correct builds. This ticket points out that the developer may forget to read in a new or edited build_script when mbuild is invoked.

In typical mbuild changes, the developer may hand-type a build_script file; or copy a build script file from one installation directory to another, then edit that target file accordingly; or use mbuild's save request to fabricate a build_script file which is then edited. In any of these cases, the developer might (in real life, did) forget to read the new/edited build_script into the database when mbuild is reinvoked.

The ticket suggests that mbuild, when first scanning the installation directory, should check whether a build script exists; and if one is found, should ask the user whether that build script should be read. The prompt acts as a reminder to the mbuild user.

The ticket further gives a sample installation directory, and proposes an action:

```
ls <mbrp>**

Segments = 2, Lengths = 4.

r w    1  mbrp.mb
r w    3  mbuild_scan_.pl1

r 09:33 0.291 0

mbuild

Installation_directory: >udd>m>gd>w>mbrp

           Scanning…


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

Description:
 1) Change scan request to:
    ...
```

This enhancement may be summarized as follows:

A. When mbuild is invoked, its default start-up actions should:

- scan the installation directory, looking for segments to build/install;
- if a build script exists, ask if that script should be read into the mbuild database.

Control arguments for the mbuild command should permit the user to avoid the initial scan, or the prompt to read an existing build script.

When implementing this enhancement, several other mbuild issues, bugs, or enhancements were identified. These are summarized below, and are included as part of this proposal.

B. If a build script has several names, mbuild may describe the file with a Seg.name value that does not match the name for this mbuild instance. mbuild uses the shortest name on the installation directory as its instance name: the *installation name*. This uncertainty about the name by which mbuild knows the build script makes it difficult for mbuild to search for that script in its internal database.

C. When searching for segments in the mbuild database, mbuild's seek code should check all added names on each segment. Currently, search checks only the primary name by which mbuild knows the segment (its Seg.name value). This suggestion to seek against all segment names applies: to searches performed by mbuild during its scan and analysis requests; and to user requests such as compile, compare, and history that select individual files on which to operate. Names for structures in mbuild's database should mirror the real file system entries they represent. Otherwise, mbuild's modeling within its database fails to match its user's view of the installation directory.

D. If a build script exists when mbuild is invoked, mbuild should not try to overwrite this script with new data until it has sufficient knowledge to write a complete view of the proposed change components. In particular, output of the analyze request is needed to write a correct and complete build specification. Saving into the build script before analyze puts data in that file that cannot be handled by a read request; read rejects such data as being improperly formatted.

E. When issuing a set -description request, the description entered should be savable in an existing build script segment no matter how far the build process has proceeded. The description should be savable even prior to an analysis request.

F. If problems D and E are resolved, it would then be proper for the set -description request to automatically store each newly-entered description into the build script, creating a new build script file if one does not already exist. Users have requested this feature (informally, not via a ticket).

G. When a clean request is issued, the request needs output from the analyze request to determine which segments in the installation directory are candidates for deletion. So clean request automatically calls the analyze request, if one has not been performed earlier. However, the analyze request unconditionally calls the progress request after a successful analysis; and such progress information is not appropriate as part of the clean output. The clean request lists candidate segments for deletion, then asks the user to ok deleting the candidates listed above. This question becomes confusing when a progress report is included seemingly as part of the candidate list. There should be some way to use an analyze request without invoking the progress request.

H. When changing and testing a bound segment having many components, it can become difficult to determine which components in the developer's directory have actually been changed. mbuild should provide a request for listing original-content segments in a development directory that differ from their corresponding library versions. This would aid in tracking which components have been modified as part of the current change. These are the segments that actually need to be installed to implement the change.

I. The scan request sometimes misreports an original-content source file as being an intermediate source file (a file derived from the first step of a 2-step translation process). A problem in the algorithm for early identification of these intermediate segments causes this reporting anomaly.

Proposed Changes

Enhancement (A): change the mbuild scan request (mbuild_scan_.pl1) to look for a build script file in the installation directory. If found, scan will ask the user if that build_script file should be read into the mbuild database.

The proposed change is an introductory header, followed by a question naming the build_script file:

```
Scanning, with read of any build_script...
```

mbuild (scan): **Do you want to read the mbrp.mb file?**

The user may answer:

yes or **y**: to automatically invoke the read request;

no or **n**: to delay reading the build_script to allow other requests to be issued first; or

?: to display more information about the prompt.

If **?** is the response, a more explanatory prompt is issued:

mbuild (scan): Do you want to read the mbrp.mb file? **?**

Read into mbuild the existing build_script found in installation directory?

This prompt is displayed only during the first use of the scan request. Any subsequent scan during that mbuild subsystem invocation does not issue the prompt.

The mbuild command (invoking the subsystem) already includes -scan and -no_scan control arguments controlling its pre-request-loop actions. With this proposal, a -scan_read control is added to prompt for reading of the existing build script. These three control arguments select from one of three mbuild start-up options:

```
help mbuild -ca scan
2020-01-25  mbuild, mb

Control arguments:
-scan_read, -scrd
   perform a scan of the installation directory before entering the
   request loop.  If a build script segment is found, ask the user
   if it should be read.  (default)
-scan, -sc
   perform a scan of the installation directory before entering the
   request loop.  Do not ask about, or actually read, any build script
   file.
-no_scan, -nsc
   omit the scan request before entering the request loop.
```

To implement mbuild -scan_read, add two new control arguments to the scan request: -query and -no_query.

```
mbuild:  help scan -ca query
2020-01-25  scan, sc

Control arguments:
-query, -qy
   for the first scan in an mbuild invocation, check for a
   build script file in the installation directory.  If one is found,
   ask whether this file should be read following the scan operation.
   (default)
-no_query, -nqy
   do not ask about, or actually read, any build script file.
```

The mbuild read request already has a -brief control argument that: suppresses display of data read from the build_script, if such file exists; and does not report an error if a build script file is not found. Changes to the mbuild command (above) requires separate control args for these two functions:

```
mbuild:  help read -all
2020-01-26  read, rd
 . . .
Control arguments:
 . . .
-print, -pr
   Print segment information obtained by this read request. (default)
-no_print, -npr
   Do not print data read by this request.
-brief, -bf
   Do not report an error if the build script file does not exist.
```

Problem (B): for entries found by the scan request, scan calls hcs_$star to obtain entry names and type of each segment in the installation directory. By default, it uses the longest name on a segment as the primary name by which that segment is known in the mbuild database: its Seg.name value. But for some segment types, knowledge of Multics Libraries enables a better choice of primary name.

- For a bound segment's source and object archives and bound object, scan uses the first name on the segment if that name begins with: bound_, because such name and location in the name list are typical identifiers for the bound segment.
- For info segments, it uses the first name on the segment, presuming names have been ordered by the developer or installer to be in the most meaningful order.

For the build script, this proposal makes the scan request use the name matching the installation name. (mbuild uses the shortest name on the installation directory as the installation name.)

Thus, if the installation directory has names of: mbuild_enhance_01 and mb01, the installation is named mb01 in the mbuild database. So scan should look for a build_script named: mb01.mb

Change the scan request to select this name as the Seg.name in the build script's Seg structure.

Problem (C): the mbuild scan request uses mbuild_Tlist_$find_Seg to locate a Seg structure matching a particular name in the Seg list built by this request.

Change mbuild_Tlist_.pl1 as follows:

- change the $find_Seg entrypoint to compare against both Seg.name and Seg.added_nameP elements when searching for Seg having a given name.
- add $request_COMPILE_match_star function to search for a COMPILE structure in a request list that matches a given starname. COMPILE structures only have a COMPILE.name element, which is set to the COMPILE.sourceP -> Seg.name value.
- add $request_Seg_match_star functions to search for a Seg structure in a request list that matches a given starname. The starname is compared against the Seg.name and Seg.added_nameP elements.

Change the mbuild compile request (mbuild_compile_.pl1) to use the new $request_COMPILE_match_star function to implement its SEG_NAME search.

Change the compare (mbuild_compare_.pl1) and history (mbuild_history_.pl1) requests to use the new $request_Seg_match_star function to implement their SEG_NAME search.

Problem (D) and (E): change the mbuild save request (mbuild_script_.pl1) to add -description and -all control arguments:

```
Control arguments:
-all, -a
   save complete information about the current build operation.
   (default)
-description, -desc
   save only the description set for this installation into the
   Description field of the build script segment.  Any other content
   of the segment is unchanged.  This control may be given to save the
   description before an analyze request has successfully completed.
```

The save request uses a file_output command followed by a print request to completely replace an existing build_script, or create a new build script file. Change the print request (mbuild_print_.pl1) -description, -analyze, and -save_format control arguments to select which parts of the build_script file should be print/saved.

The revised descriptions for these control arguments are:

```
List of Information Types:
-description, -desc
   print the current description added by the installer for this
   installation directory.  The description is specified using set
   request, or by editing the Build Script file to add/change the
   Description: lines at the top of this file.
-analyze, -az
   print installation data reshaped by an analyze request.  (default
   after an analyze request is issued.)
-save_format, -save
   print installation data for the -description or -analyze arguments
   as it would appear in the build script segment.
```

Change the save request to correctly invoke the print request with revised control arguments when adding/replacing the entire build script file. This includes the cases:

• save -description, when the build_script does not currently exist.

• save -all, whether or not the build_script currently exists.

If save -description is given when a build script file currently exists, provide code in the save request to add/replace just the Description section of that file. Other parts of that file are not changed.

Enhancement (F): Change the set -description request (mbuild_set_.pl1) to automatically call the save -description request after it stores a new description value into the mbuild database.

Problem (G): Change the clean request (mbuild_clean_.pl1) to invoke the analyze request (when needed by clean) with a new -no_progress control argument.

Change the analyze request (mbuild_analyze_.pl1) to support that new -no_progress control argument:

```
mbuild:  help analyze -ca progress
2020-01-26  analyze, az

Control arguments:
-progress, -pg
   Run the progress request when analysis completes successfully.
   (default)
-no_progress, -npg
   Omit the progress request when analysis completes successfully.
```

Enhancement (H): Change the compare request (mbuild_compare_.pl1) to add a -brief control argument.

The revised description for the -brief control is as follows:

```
mbuild:  help compare -ca brief
2020-01-26  compare, cmp

Control arguments:
-brief, -bf
   use the Multics compare active function to identify replaced
   segments whose contents differs from their corresponding library
   version.  Only the file name and corresponding library path is
   displayed.
```

Sample output of the compare -brief request is shown below:

```
mbuild:  cmp -bf
 Segments differ from their library counterparts:

  mbuild.pl1            vs.  >ldd>tools>source>bound_mbuild_.s::mbuild.pl1
  mbuild_Tlist_.pl1     vs.  >ldd>tools>source>bound_mbuild_.s::mbuild_Tlist_.pl1
  mbuild_analyze_.pl1   vs.  >ldd>tools>source>bound_mbuild_.s::mbuild_analyze_.pl1
  mbuild_clean_.pl1     vs.  >ldd>tools>source>bound_mbuild_.s::mbuild_clean_.pl1
  mbuild_compare_.pl1   vs.  >ldd>tools>source>bound_mbuild_.s::mbuild_compare_.pl1
```

This change adds the mbuild_library_$compare function to invoke the Multics compare active function for a given pair of files, and return true if content of the two files differs. This function is used by the compare request to implement its new -brief control argument.

Problem (I): When an enhancement for the scan request was suggested (A above), a small bug was also mentioned. The scan request sometimes pointed out a source file as an intermediate source (one generated by the first step of a 2-step translation process). This notice was incorrect; the named source was an original-content file, and was not part of a 2-step translation.

No ticket was ever entered for this bug, since it seemed related the forgotten read of the build_script. However, analysis of mbuild_scan_.pl1 for the prompt enhancement also discovered a code defect causing this mistaken informational message. That defect is repaired as part of this change proposal.

Components of the Change

The build_script for this proposed change describes files to be modified.

```
Description:
 1) Change scan request to:
     - prompt installer to read the build_script file if one is found
       in the installation directory. [Ticket #196]
     - use build_script name that matches the installation directory,
       for a build_script having several names.
     - fix algorithm for identifying intermediate file's in a 2-step
       translation process (e.g., .cds -> .pl1 -> object_seg).
 2) Change mbuild_Tlist_ to:
     - make $find_Seg search for the desired Seg name, looking also
       in the Seg's add_name list.  This helps implement (1) above.
     - add $request_Seg_match_star to search for the desired Seg name,
       looking also in the Seg's add_name list.  This helps implement
       (9) and (10) below.
```

```
 3) Change mbuild command to add -scan_read control argument, which follows
    an automatic scan of installation directory by prompting to read any
    build_script segment.  This becomes the default behavior for mbuild
    invocations.
 4) Change save request to:
     - add a -description control argument which saves just the Description
       field in an existing or new build script segment.  This helps implement
       (11) below.
 5) Change print request to:
     - implement (4) above.  This involves slight changes to the print request's
       control arguments used by the save request.
 6) Add mbuild_et_$no_description error code.  This is needed to implement (4)
    above.
 7) Change clean request to:
     - invoke analyze with -no_progress, if analysis is needed to enable cleaning.
       The progress messages confuse subsequent clean output and user prompts.
 8) Change analyze request to:
     - add a -no_progress control argument that omits calling progress request
       after a successful analysis.  This is needed to implement (7) above.
 9) Change compare request to:
     - add -brief control argument that lists names of REPLACEd segments whose
       contents differ from their library counterpart.
     - compare SEG_NAME argument with names on Seg.added_namesP list.
10) Change history request to:
     - compare SEG_NAME argument with names on Seg.added_namesP list.
11) Change set -description request to:
     - end setting the new description by invoking request: save -description.
12) Add mbuild_library_$compare function, to implement (9) above.
13) Add mbuild_info_find_$prefix_for_build_type function, to implement (1) above.

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

 Build_script:            mbrp.mb;

 Bound_obj:               bound_mbuild_          IN: tools  UPDATE;
     source:              mbuild.pl1              REPLACE           compiler: pl1 -ot;
     source:              mbuild_Tlist_.pl1       REPLACE           compiler: pl1 -ot;
     source:              mbuild_analyze_.pl1     REPLACE           compiler: pl1 -ot;
     source:              mbuild_clean_.pl1       REPLACE           compiler: pl1 -ot;
     source:              mbuild_compare_.pl1     REPLACE           compiler: pl1 -ot;
     source:              mbuild_et_.alm          REPLACE           compiler: alm;
     source:              mbuild_history_.pl1     REPLACE           compiler: pl1 -ot;
     source:              mbuild_info_find_.pl1   REPLACE           compiler: pl1 -ot;
     source:              mbuild_library_.pl1     REPLACE           compiler: pl1 -ot;
     source:              mbuild_print_.pl1       REPLACE           compiler: pl1 -ot;
     source:              mbuild_scan_.pl1        REPLACE           compiler: pl1 -ot;
     source:              mbuild_script_.pl1      REPLACE           compiler: pl1 -ot;
     source:              mbuild_set_.pl1         REPLACE           compiler: pl1 -ot;

 Info:                    mbuild.info           IN: priv.info  REPLACE;
                            add_name:
                              mb.info
                              build_script.gi.info;
```

Testing

Direct testing of these changes has followed three paradigms.

First, as each new feature was added, it was incorporated into a test version of mbuild. Individual sources were compiled with symbol tables and run as unbound segments. When/if problems were observed, full probe debugging techniques could quickly isolate the defective design or implementation. These tests involved installing changed mbuild components into a private library; and verifying both operation of mbuild and results in using the end product of the build.

Second, the private library version of mbuild was kept up-to-date with changes outlined in this proposal. It was then tested against the original mbuild test suite (described in MTB-1003).

Third, my auditor Eric Swenson plans to use the revised code in preparing some of the Multics changes planned for the MR12.7 release. This effort is currently underway, and will involve replacing many source modules and bound segments in the Multics Libraries.

If testing and code audits find no further problems, the mbuild version described in this MCR will be installed into the Multics Libraries, and used officially to build/install the final MR12.7 changes.

Documentation

The changes to the mbuild command and its requests were briefly described above. Documentation for those changes is given here, as changes to the mbuild.info segment. That segment includes documentation for the mbuild command interfaces, and for most of its subsystem requests.

The requests not documented in mbuild.info are standard ssu_ requests, which were unchanged by this proposal.

Changes to the mbuild command:

```
mbuild:  cmp mb.info -minlines 5

----------  mbuild.info

compare_ascii >doc>privileged>mbuild.info == -minlines 5

A1        :Info: mbuild: mb:  2019-10-26  mbuild, mb
Changed by B to:
B1        :Info: mbuild: mb:  2020-01-27  mbuild, mb


Inserted in B:
B44       -scan_read, -scrd
B45          perform a scan of the installation directory before entering the
B46          request loop.  If a build script file is found, ask the user
B47          if it should be read.  (default)
B48       -scan, -sc
B49          perform a scan of the installation directory before entering the
B50          request loop.  Do not ask about, or actually read, any build script
B51          file.
B52       -no_scan, -nsc
B53          omit the scan request before entering the request loop.
B54
B55
Preceding:
A44       -prompt STR, -pmt STR


A49       -scan, -sc
A50          perform a scan request of installation directory before entering
A51          request loop. (default)
A52       -no_scan, -nsc
A53          omit scan request before entering the request loop.
A54
Changed by B to:
B61
```

Changes to the scan request:

```
A95       :Info: scan: sc:  2019-07-13  scan, sc
A96
A97       Syntax as a request:  scan
Changed by B to:
B102      :Info: scan: sc:  2020-01-25  scan, sc
B103
B104      Syntax as a request:  scan {-control_arg}...


A104      :Info: read: rd:  2019-10-26  read, rd
Changed by B to:
B111      Control arguments:
B112      -query, -qy
B113         for the first scan in an mbuild invocation, check for a
B114         build script file in the installation directory.  If one is found,
B115         ask whether this file should be read following the scan operation.
B116         (default)
B117      -no_query, -nqy
B118         do not ask about, or actually read, any build script file.
B119
B120
```

Changes to the read request:

```
B121      :Info: read: rd:  2020-01-26  read, rd


A127      -brief, -bf
A128         Do not print segment information obtained by this read request.
A129         Do not report an error if the segment does not exist.
Changed by B to:
B144      -no_print, -npr
B145         Do not print data read by this request.
B146      -brief, -bf
B147         Do not report an error if the build script file does not exist.
```

Changes to the analyze request:

```
A140      :Info: analyze: az:  2019-10-26  analyze, az
Changed by B to:
B158      :Info: analyze: az:  2020-01-26  analyze, az


A158
A159
A160      :Info: print: pr: p:  2019-10-23  print, pr, p
Changed by B to:
B176      -progress, -pg
B177         Run the progress request when analysis completes successfully.
B178         (default)
B179      -no_progress, -npg
B180         Omit the progress request when analysis completes successfully.
B181
B182
B183      :Info: print: pr: p:  2019-01-29  print, pr, p
```

Changes to the print request:

```
A206      -save_format, -save
A207         print installation data as it would look in the build script file.
Deleted by B, preceding:
B229      -Seg ENTRYNAME, -seg ENTRYNAME


Inserted in B:
B238      -save_format, -save
B239         print installation data for the -description or -analyze arguments
B240         as it would appear in the build script file.  With -analyze, the
B241         description is displayed if a description has been set.
Preceding:
A217
A218
A219      Control arguments:
```

Changes to the set request:

```
A351      :Info: set:  2019-10-23  set
Changed by B to:
B376      :Info: set:  2020-01-28  set


A362      After setting a new description, use the save request to capture that
A363      description at the top of the build script file.  This permits it to be
A364      saved/used across invocations of mbuild.
A365
A366
A367      Control arguments:
A368      -description, -desc
A369         prompt for a new description of the installation.  The description
A370         ends with a line containing only a period (.) character.
Changed by B to:
B387
B388      Control arguments:
B389      -description, -desc
B390         prompts for a new description of the installation.  The description
B391         ends with a line containing only a period (.) character.
B392         The new description value is saved in a build_script file to
B393         preserve this setting across mbuild invocations.
```

Changes to the save and compare requests:

```
A388      :Info: save: sv:  2019-07-13  save, sv
A389
A390      Syntax as a request:  sv
A391
A392      Function: saves information about the current build operation to a
A393      build script file.
A394
A395
A396      :Info: compare: cmp:  2019-10-23  compare, cmp
A397
A398      Syntax as a request:  cmp {SEG_NAME} {-cpa_control_arg}...
Changed by B to:
B411      :Info: save: sv:  2020-01-25  save, sv
B412
B413      Syntax as a request:  sv {-control_arg}
B414
B415      Function: save information about the current build operation to a
B416      build script file.  In order to save full build information, the
B417      analyze request must have been run since the last scan or read
B418      request.
B419
B420
B421      Control arguments:
B422      -all, -a
B423         save complete information about the current build operation.
B424         (default)
B425      -description, -desc
B426         save only the description set for this installation into the
B427         Description field of the build script file.  Any other content
B428         of the segment is unchanged.  This control may be given to save the
B429         description before an analyze request has successfully completed.
B430
B431
B432      :Info: compare: cmp:  2020-01-26  compare, cmp
B433
B434      Syntax as a request:  cmp {SEG_NAME} {-cpa_control_arg}...
B435                            cmp {SEG_NAME} {-brief}


Inserted in B:
B457      -brief, -bf
B458         use the Multics compare active function to identify replaced
B459         segments whose contents differs from their corresponding library
B460         version.  Only the file name and corresponding library path is
B461         displayed.
B462
Preceding:
A420
A421      :Info: history: hcom:  2019-10-23  history, hcom
```

Changes to the compile request:

```
A709      :Info: compile: comp:  2019-09-09  compile, comp
Changed by B to:
B752      :Info: compile: comp:  2019-01-29  compile, comp


A723         after correcting a compilation error.  The star convention is not
Changed by B to:
B766         after correcting a compilation error.  The star convention is


A791         Build_script file (created by an earlier save request).  Used when
A792         starting over on a build effort when compile or archive_prep
Changed by B to:
B834         Build_script file (created by an earlier save request).  Used
B835         when starting over on a build effort when compile or archive_prep
```

Changes to the mbuild.info history comment:

```
A1098     :Internal: ~history_comment:  2019-08-17  history_comment
Changed by B to:
B1141     :Internal: history_comment.gi:  2020-01-25  history_comment


A1105                                                      END HISTORY COMMENTS */
Changed by B to:
B1148       2) change(2020-01-25,GDixon), approve(2020-01-25,MCR10079):
B1149          A) To mbuild command, add control argument: -scan_read (default)
B1150          B) To scan request, add control arguments: -query and -no_query
B1151          C) To save request, add control argument: -description
B1152          D) To analyze request, add control argument: -no_progress
B1153          E) To read request, add control argument: -no_print
B1154          F) To compare request, add control argument: -brief
B1155          G) To set request, noting that -description now saves the new
B1156             description setting to a build_script file.
B1157          H) To print request, refine how -save_format works with
B1158             -description and -analyze control arguments.
B1159                                                      END HISTORY COMMENTS */

Comparison finished: 19 differences, 148 lines.
```

Version History

Date	Revision	Author	Comment
2020-01-30	1.0	Gary Dixon	Initial version of this MCR.