Subject: MCR10161, *cross_reference* Command Generates Malformed Source Program Names

Author: Gary Dixon

Date: March 9, 2026

Multics Ticket: http://multics-trac.swenson.org/ticket/386

Multics Ticket 386 documents an issue concerning data produced by the Multics cross_reference (cref) command. The resulting .crossref output segment includes entries corresponding to each object program or source include segment that is cross-referenced. Each entry specifies the source program(s) that either reference an external entry point within the analyzed source file or include the specified include segment.

Analysis

The expected .crossref output should have the format:

```
--------------------                ***** bound_chaos_file_ in NETWORK *****
cftp_request_table_
 cftp_request_table_
                    chaos_ftp.pl1
```

However, the cref command sometimes formats names of referencing programs like an *archive-pathname* entry name, as shown below.[1]

The reference name `chaos_ftp.s::chaos_ftp.pl1` is malformed. The archive part does not identify an actual segment: chaos_ftp.s.archive.

```
--------------------                ***** bound_chaos_file_ in NETWORK *****
cftp_request_table_
 cftp_request_table_
                    chaos_ftp.s::chaos_ftp.pl1
```

[1] In an archive pathname, :: separates the archive name from the stored component. For example, bound_crossref_.s::cross_reference.pl1 refers to the cross_reference.pl1 component in the bound_crossref_.s.archive.

The malformed names can grow so long that important parts of the name get truncated. Several of the entry names below are missing the language suffix and other characters.

```
--------------------                ***** bound_chaos_file_ in NETWORK *****
chaos_ftp_data_convert_
 bitcount_to_filepos
                    mgp_chfile_server_.s::mgp_chfile
 chaos_ftp_data_convert_
                    cftp_.s::cftp_.pl1  mgp_chfile_server_.s::mgp_chfile
 filepos_to_bitcount
                    mgp_chfile_server_.s::mgp_chfile
 print              cftp_.s::cftp_.pl1
```

Background Details

The cross_reference command calls the cref_analyze_.pl1 source program to gather cross-reference data from a given bound object segment. cref_analyze_.pl1 examines object components bound within that object segment, extracting:

- pathname of the source program compiled to produce that object component;
- pathname(s) of include segments referenced by that source program;
- names of external entry points into the component which are retained by the bound object segment; and
- names of subroutine references linking the object component to external entry points in other programs.

The symbol table information in each object component contains the pathname of its source program and include segments. The language suffix (.pl1, .lisp, .alm, etc.) specifies the programming language in which that source file is coded. cref_analyze_.pl1 thereby learns the rules used by that language object to link to external program entry points and data; and the external entry points retained into the object segment which other programs may reference.

Problem Analysis

The first example above shows an issue in formatting information about the cftp_request_table_ object segment. The data below shows the source program pathname for cftp_request_table_, an object component of bound_chaos_file_ object segment; and the include segment pathnames included by that source program. These are the first items obtained when cref_analyze_.pl1 examines an object component.

```
get_component_info >net>bound_chaos_file_ cftp_request_table_ source
>udd>sa>ejs>w>mgp_installs>003>bound_chaos_file_.s::cftp_request_table_.alm
>ldd>include>ssu_request_macros.incl.alm
>ldd>include>stack_header.incl.alm
```

Notice that the source for this program is shown as an archive pathname. A program installed into the Multics system libraries is usually compiled from a standalone source segment. In this case, the source resided in an archive component.

When the mbuild IDE[2] subsystem was created, the build process changed in some ways. mbuild requires source programs for a new bound segment to be stored in their new source archive. This tells mbuild which new bound segment will hold the compiled object from each source.

When mbuild compiles a source for a new bound segment, the source resides in the source archive. Thus, the source pathname stored in the compile object is an archive pathname. Perhaps the cross_reference problem is an unexpected side effect of this change to the build process.

[2] IDE stands for Integrated Development Environment. It is a software application that combines essential developer tools—such as a code editor, compiler, and build automation—into a single subsystem to streamline programming and increase productivity.

Root Cause

After Eric Swenson created Ticket_386, he did some investigation into of the cref_analyze_.pl1 program and found that the following code references the source pathname for each object, but it is not prepared to split an archive pathname into its directory, archive name, and archive component parts.

```
417   /* Get language suffix if available (some translators like lisp skip it) */
418
419         lang_suffix_node = null;
420
421         if oi.source_map > 0 then do;
422               source_map_p = addrel (oi.symbp, oi.source_map);
423               source_path_ptr = addrel (oi.symbp,
424                                         source_map_p -> source_map.offset (1));
425               source_path_len =
426                    binary (source_map_p -> source_map.size (1));
427               i = source_path_len + 2
428                    - index (reverse (source_path), ">");
429               j = index (substr (source_path, i), ".");
430               if j >= 0 then
431                     lang_suffix_node = cref_listman_$create_environment
432                                         (substr (source_path, i + j - 1), "0"b);
```

Eric suspected this code might be causing the malformed entry names. However, it was not immediately obvious how code that obtains source program pathnames could affect code elsewhere that formats source program names.

When I changed the suspect code to use expand_pathname_$component to properly separate the parts of an archive pathname, then no malformed entry names were generated. So it appears Eric's suspicion was correct.

```
431   /* Get language suffix if available (some translators like lisp skip it) */
432
433         lang_suffix_node = null;
434
435         if oi.source_map > 0 then do;
436               source_map_p = addrel (oi.symbp, oi.source_map);
437               source_path_ptr = addrel (oi.symbp,
438                                         source_map_p -> source_map.offset (1));
439               source_path_len =
440                    binary (source_map_p -> source_map.size (1));
441
442  dcl  dir  char(168), /* Split possible archive pathname of source program      */
443       ent  char(32),  /*   to get the source's entryname.  Extract language     */
444       comp char(32);  /*   suffix (e.g., ".pl1") from either comp or ent.       */
445
446               call expand_pathname_$component( source_path, dir, ent, comp, code );
447               if  comp ^= ""  then
448                     ent = comp;
449               j = index( ent, ".");
450               if  j > 0  then
451                   lang_suffix_node = cref_listman_$create_environment
                                           ( substr(ent, j), NO_BINDFILE );
```

Proposed Changes

Modernize cref_analyze_ as follows:

A. Call expand_pathname_$component to split source pathname (from object program's list of source inputs) to get a language suffix for the cref language_suffix_node.

B. Use the documented archive_$next_component to scan archive components. The archive_util_ subroutines currently used by cref_analyze_ are obsolete interfaces. Their callers must know too much about the internal organization of archive segments.

```
Changes --

 Bound_obj:    bound_crossref_                  IN: tools  UPDATE;
   source:     cref_analyze_.pl1                 REPLACE  compiler: pl1 -ot;
```

Testing

These changes were tested by creating cross-reference data for the proposed new network library: first with the installed cross_reference command; then with the repaired cross_reference command. Output of the former was examined: many instances of names with two consecutive colon characters were found. In output generated by the repaired cross_reference, no instances of double-colon were found.

Documentation

The above changes do not alter the user interface for the cross_reference (cref) command, so no documentation changes are needed.

Version History

Date	Revision	Author	Comment
2026-03-09	1.0	Gary Dixon	Initial Draft
2026-03-10	1.1	Gary Dixon	Revised repair of cref_analyse_.pl1.