Subject: MCR-10031, Command/AF/Subroutine Interface to interpret_ptr_

Author: Gary Dixon

Date: February 18, 2017

The interpret_ptr_ subroutine does an excellent job of looking up an entrypoint name in bound object segments, and displaying information about them. Given 243|3734, it returns information about process_overseer_$mme2_fault_handler_. It is often useful to quickly do such lookups as a command or active function.

Also, interpret_ptr_ was designed as a support routine for trace_stack. Its interface returns a structure of information about the pointer being investigated. It is not immediately obvious to an external caller how to make immediate use of this information. A wrapper subroutine could provide a simpler interface to this useful subroutine.

- Reference URL of Multics Change Ticket: http://multics-trac.swenson.org/ticket/46

Proposed Changes

An enhancement to provide a command, active function, and subroutine interface to interpret_ptr_ in a single source. The command would be called: pointer_info; with short name: pin. The subroutine interfaces would be: pointer_info_; and pointer_info_$location. Proposed user interfaces are provided in the documentation section of this MCR.

• Add source:	pointer_info.pl1
• To bound segment:	>sss>bound_trace_stack_
• Change bind file:	add synonyms on pointer_info: pin, pointer_info_ retain: pointer_info, pin, pointer_info_, pointer_info_$location
• Add info segments:	>doc>info>pointer_info.info (pin.info); >doc>info>pointer_info_.info

Documentation

Documentation for the pointer_info command/active function:

```
02/17/2017  pointer_info, pin

Syntax as a command:  pin virtual_pointer {-control_args}

Syntax as an active function:  [pin virtual_pointer {-control_args}]

Function: displays information about a pointer value -

  reference_name$entryname pointed to;
  objectname$entryname or objectname$offset within a bound segment;
  ring_0_segment|offset for an inner-ring pointer.

Additional information about the pointer value may be displayed with
the pointer_info command.

Arguments:
virtual_ptr
   is a character string representing the pointer value to be
   interpreted.  For a list of accepted character representations,
   see: virtual_pointers.gi.info.
-location virtual_ptr, -loc virtual_ptr
   is a character string representing the location of the pointer
   storage to be interpreted.  This form is useful when you know
   where the pointer is stored, rather than its value.  For example,
   when interpreting an unsnapped link in the linkage section of an
   object segment, giving the location of this link provides clues to
   obtaining its link definition.

Control arguments:
-all, -a
   the command displays additional information returned by the
   interpret_ptr_ subroutine regarding the pointer.  Normally, only
   a brief interpretation of the pointer is displayed by the command,
   or returned by the active function.

Notes:
The pointer value must reside at an even-word location, and include
either an ITS modifier (octal 43) or a Fault_Tag_2 (unsnapped link,
octal 46) modifier ending the first word of the pointer word pair.
The interpret_ptr_ subroutine verifies these requirements before
interpreting the pointer value.

List of examples:
When dumping contents of an object segment, you wish to investigate a
pair of words that looks like an unsnapped link points, to learn what
the snapped link would reference.

 dump_segment <tests>hello 100 20
 000100 000000000000 000000000000 000000000000 000045000000
 000104 000000000000 000000000000 000000000000 000000000000
 000110 000010000014 000000000000 777770000046 000021000000
 000114 777766000046 000027000000 000000000001 163171155142
 r 21:41 0.057 0
```

Offset 114 in this file is a word ending with the unsnapped link
modifier (octal 46). To ask for information about this possible
pointer.

```
 pin -loc <tests>hello|114
  For pointer: 77766|27
    information:     ioa_$nnl
 r 21:42 0.031 0
```

Use -all to obtain more information.

```
 pin -loc <tests>hello|114 -all
  For pointer: 77766|27
    information:     ioa_$nnl

    octal pointer:   777766000046 000027000000
    comment:         (unsnapped link)
    segment:         ioa_
    entryn:          $nnl
 r 21:42 0.051 0
```

Obtain information about segment 75.

```
 pin 75|0
  For pointer: 75|0
    information:     restart_fault$0 (ring 0)
 r 21:57 0.053 0
```

Obtain information about the fault_vector segment.

```
 pin fault_vector$0
  For pointer: 4|0
    information:     fault_vector$0 (ring 0)
 r 21:59 0.053 0
```

Documentation for the two entrypoints of the pointer_info_ function:

```
02/17/2017  pointer_info_

Functions to interpret pointer values.

Entry points in pointer_info_:
02/17/2017  pointer_info_
02/17/2017  pointer_info_$location


Entry:  02/17/2017  pointer_info_   (11 lines in entry point)

Function:
This function returns information about a pointer value.


Syntax:
dcl pointer_info_ entry (ptr, fixed bin(35)) returns(char(76) var);
data = pointer_info_(pointer_value, code);


Arguments:
pointer_value
   is an aligned pointer value to be investigated.(Input)
code
   is a standard status code.(Output)  If nonzero, then an empty
   string is returned as the data.




Entry:  02/17/2017  pointer_info_$location   (19 lines in entry point)

Function:
This function returns information about pointer storage (a pair of
words) at a specified location.  It checks for a pointer modifier
at the end of the first word.


Syntax:
dcl pointer_info_$location entry (ptr, fixed bin(35))
       returns(char(76) var);
data = pointer_info_$location(pointer_loc, code);


Arguments:
pointer_loc
   is an aligned pointer to the storage location containing the
   pointer to be investigated.  This storage must begin on an even
   word boundary.  The pair of words at that location are
   investigated as a possible pointer value, having either an ITS
   (octal 43) or Fault Tag 2 (octal 46) modifier at the end of the
   first word in the pair.
code
   is a standard status code.(Output)  If nonzero, then an empty
   string is returned as the data.
```

Version History

Date	Revision	Author	Comment
2017-02-18	0.1	Gary Dixon	Initial draft.