TO: MSPM Distribution

FROM: C. Garman

DATE: January 6, 1969

SUBJECT: get_sym_def, BY.13.05

Attached is the description of get_sym_def, whose usage is encouraged for circumstances where linkage definitions must be searched, but where the value required is to be used only temporarily and for data-access only.

Identification

Search linkage definitions for external symbol
get_sym_def
Charles Garman

Purpose

The purpose of get_sym_def is to search the definition information of a segment for a specific external symbol, and to return a pointer to the defined location in either the text, link, or symbol section (as determined by the "class" of the definition). It is primarily intended for use by the Binder.

Usage

```
dcl get_sym_def ext entry returns (ptr),
     (p, textp, linkp, symbp) ptr,
     symbol char(K);
p=get_sym_def (symbol, textp, linkp, symbp);
```

symbol is character string whose value is sought in the linkage definitions of the set of "related" text, link and symbol segments pointed to by textp, linkp, and symbp respectively (linkp may point to the start of the copy of the linkage section in the combined linkage segment), and p is the value returned by get_sym_def.

(In the declarations above, K is used to indicate an arbitrary length string, $0 \leq K \leq 31$; symbol may also be a character string constant in the actual call, eg. "rel_text".)

Implementation

On entry to get_sym_def, the length of symbol is determined: if the length is zero, the value returned is that of textp; if the length is > 31, the null pointer is returned.

Next, a copy of the string is made in such a form that a direct word-by-word comparison may be made later: The first character of the first word contains the count in binary, the characters of symbol are copied in starting at the 2nd character in the word and trailing character positions are filled with binary zeroes. (This produces a sequence of words in ACC format-ASCII Characters with Count.)

Next begins the search of the definition block(s): the pointer linkp is assumed to point to the start of an eight-word linkage header, and a pointer to the external definitions is created (the program handles linkage sections in either the "loaded" or "unloaded" form, that is, the right-most 6 bits of the first word may be 0, or octal 20, or an ITS modifier; if it is none of these, searching stops and a null value is returned).

For each external-symbol-definition block, the class field is first examined: if it is all 1's, the program proceeds to the next chained definition. Next the number of characters of the defined symbol is compared with the number of characters of symbol, and if equal, a word-by-word comparison is then initiated: if any mismatch occurs, the examination is terminated and the program proceeds to the examine the next chained definition.

Once a symbol match has been established, one more check is necessary: if the "trap" field indicates trap-before-definition the null value is returned. Otherwise the value of the get_sym_def function is computed by first selecting the pointer indicated by the "class" field, and then combining that pointer (via the "addrel" function) with the "value" field of the definition block. The newly-computed pointer value is then returned to the user.

When a definition block is exhausted as indicated by the "next" field pointing to a word of all zeroes, the header block is examined for chained header blocks (again with checks for "loaded" or "unloaded" linkage sections); if there were no further linkage blocks, the null pointer is returned.

Notes

Under no circumstances does get_sym_def overtly modify any of the three segments to which pointers are provided, nor does it modify the states of any of the given segments (such as causing them to be made unknown).