Identification

Macro Command
K. J. Martin

Purpose

A macro is merely a segment prepared using the context
editor. In order for the macro to be handled properly
when invoked as a command, it must be recognized by the
Shell as a macro. The macro command makes the edited
segment recognizable to the Shell.

Discussion

When the Shell (BX.2.00) makes linkage to a command it
calls generate_ptr (BY.13.02) and checks the definition
class of the entry it intends to call. If that class
number is 64 the Shell calls the macro processor (BX.18.01)
instead of calling the entry to which it made linkage.

The macro command creates a linkage section for the macro
and makes an external definition for macro_name$macro_name
(where macro_name is the name of the edited macro segment)
with a class number of 64.

Usage

macro   macro_name

where macro_name is the name of a segment created using the
        editor. The segment macro_name contains command lines
        which include regular commands, macro control commands
        (described in BX.18.03-BX.18.08) and user procedures.
        It may also contain input lines designated to be read
        by a command in the macro.

Macro_name is located in the file system hierarchy in
the same manner as the file system commands locate a segment
(see BX.8.00). If the pathname macro_name starts with
">" it is assumed to be a pathname relative to the root
directory. Otherwise, the pathname macro_name is assumed
to be relative to the current working directory.

Implementation

Macro creates a segment macro_name.link in the directory
containing the segment macro_name. Macro_name.link is
created with the header information necessary for it to
pass as a linkage section and one external definition
for macro_name$macro_name with class number 64. If macro_name.
link already exists in the directory, macro informs the
user that it exists and will not be tampered with. It
then returns.

The declaration for the contents of macro_name.link is:

```
dcl 1 linkage,
    2 header,
        3 def_ptr ptr,                  /* points to linkage.
                                           ext_def*/
        3 nxt_blk_ptr ptr,              /* null */
        3 pre_blk_ptr ptr,              /* null */
        3 static_location bit (18),     /* zero */
        3 block_length bit (18),        /* length of the structure
                                           in words */
        3 segment_number bit (18),      /* zero */
        3 segment_length bit (18),      /* length of the structure
                                           in words */
    2 ext_def,
        3 nxt_ext_ptr bit (18),         /* zero */
        3 unused bit (18),              /* zero */
        3 value bit (18),               /* zero */
        3 class bit (18),               /* 64 */
        3 symbol char (N);              /* macro_name */
```

As indicated by the comments, many elements are zero or
null. No forward and backward pointers are needed; no
static storage will be needed; there is no value for macro_
name$macro_name; and the segment number of macro_name
when the macro is invoked is obviously not known at this
time.

For more information about linkage sections see BD.7.01.

After successfully creating macro_name.link, macro returns.