Published: 09/19/68

Identification

The merge_edit Command - Implementation
Edwin W. Meyer, Jr.

Purpose

This section lists the merge_edit package segments and their entry points along with a brief description of their functions. The segments are presented in binding order.

Description

FILENAME

This procedure is used to associate the entry name of an arbitrary pathname with a guaranteed unique six character file name.

```
call filename$fnam_init;

call filename$filename (seg_name, file_name);

call filename$bits (seg_name, file_bits);

dcl seg_name char (*),          /*input pathname*/

    file_name char(6),          /*returned filename*/

    file_bits bit(36);          /*gebcd equiv. of file_name*/
```

filename$fnam_init must be called prior to any other filename calls in order to create the filename storage segment in the process directory.

MERGE_EDIT

```
call merge_edit (arg1, arg2, arg3, arg4, arg5);

dcl (arg1, arg2, arg3, arg4, arg5) char (*);
```

This is the top level procedure in the command package, whose duties are:

a. To interpret the arguments passed via the shell (see BE.18.00 for command format).

b. Perform all initialization, file creation and moving.

c. Call mg_pass1 and mg_pass2 to execute the merge_edit.

MG_COMP

```
call mg_comp$comp_pass1 (n);

call mg_comp$comp_pass2;

dcl n fixed bin(17);          /*supplied argument indicating
                                type of control line to be
                                processed: 0-epl, 1-eplbsa,
                                2-bcpl, 3-tmgl, 4-entry*/
```

mg_comp is the processor for all control lines dealing with compilation; comp_pass1 stores the info in the list for the indicated type of line, and comp_pass2 puts out productions for all the compilations.

MG_DATA

This eplbsa data segment is the main storage for long ascii strings, card image and array templates, etc. It is declared "slvprc, slvacc" so as to be bindable with the rest of the merge_editor.

MG_LDLB

```
call mg_ldlb$ldlb_pass1 (n);

call mg_ldlb$putout (n, put_top);

dcl (n, put_top) fixed bin(17);
```

n	ldlb pass1	putout
0	tl line	load (tl or mk)
1	mk line	load (other)
2	li line	libe
3	ld line	fetch
4	--	deck

mg_ldlb is responsible for at least a portion of the processing of tl, mk, li, ld, fetch, and deck control lines. mg_ldlb$ldlb_pass1 processes a control line according to the supplied type code n, and stores the information away into the proper list. mg_ldlb$putout is used during pass2 to create and deliver productions from the supplied list "put_top" according to the supplied type code.

MG_PASS1

```
call mg_pass1 (f_dir, f_name, abort_cd);

dcl (f_dir,                          /*dir in which gecos file
                                       resides*/

     f_name) char (*),               /*entry name of gecos file*/

     abort_cd fixed bin(17);         /*if non-zero upon return
                                       from mg_pass1, the merge_edit
                                       will be aborted*/
```

mg_pass1 scans the first token of each control line and calls the proper handler for each recognized token to scan and process the remainder of the line. Some control lines are totally processed within mg_pass1 itself.

MG_PASS2

```
call mg_pass2 (w_dir);

dcl w_dir char(*);                   /* working directory pathname*/
```

The main duty of mg_pass2 is to invoke in proper order the routines that create productions from list structure and other data created during pass1 and insert them into the ascii and binary output segments. Some of this production creation and insertion is done within mg_pass2 itself.

MG_TLMK

```
call mg_tlmk$tlmk_pass2;
```

This routine creates and delivers productions for the tl and mk lists to the output segments.

MG_UTIL

```
call mg_util$ascii_line (char_string);

call mg_util$move_string (pntr, lth);

dcl char_string (*),          /*supplied string*/

    pntr ptr,                 /*pointer to base of char string*/

    lth fixed bin(17);        /*length of character string*/
```

Char_string (or the string defined by pntr and lth) plus a new-line character are inserted into the ascii control segment immediately following previously inserted strings.

```
call mg_util$bad_line;
```

Write out on-line a "bad line" message and the entire current line of the gecos control segment.

```
call mg_util$fstate (pathname, fnd_code);

dcl pathname char (*),

    fnd_code fixed bin(17);
```

Look up the directory entry defined by pathname and return with fnd_code = 0 if it exists as a non-directory branch. Otherwise fnd_code = 1.

MG_TABLES (not bound with merge_editor)

mg_tables is the data segment that contains the merge_edit tables. This segment is used by the tape daemon, not the merge_editor, but is associated with it because it is vital to the production of a proper merge_edit tape.

mg_tables contains several numbered merge_edit tables starting at 1. Each table is an ascii string left adjusted in the first word of the table and with any unused character positions in the last word of the table filled with NUL characters. A lookup table at the head of mg_tables has the property such that the right half of word j of mg_tables contains the starting location of merge_edit table j. If table j does not exist, word j = 0. Word 0 contains the number of the highest numbered merge_edit table within mg_tables. (Figure 1 illustrates the format of mg_tables.)

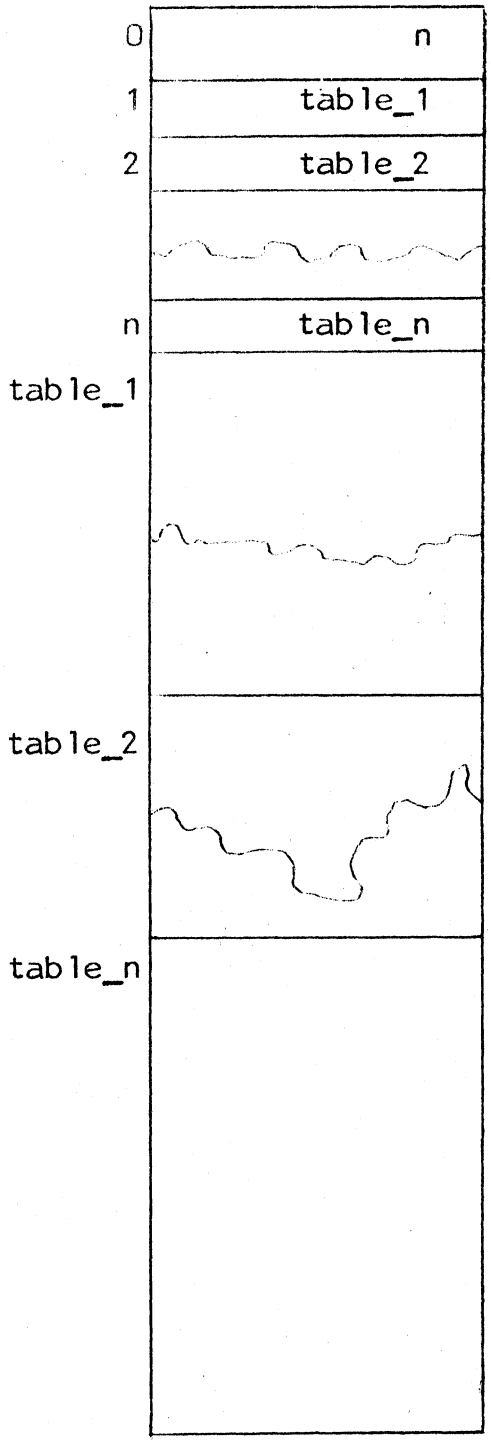

Figure 1. Format of mg_tables