Published: 05/23/68

Identification

expand_seg
S. L. Rosenbaum

Purpose

The library procedure expand_seg is a procedure which expands an input segment by inserting specified segments. It creates an output segment composed of the original input segment and its included segments. This output segment has the same entry name as the input segment with the addition of the character string ".expanded" as its last component.

The procedure expand_seg has two entry points, expand_seg and expand_seg$percent. The first entry scans the input segment for statements of the form

"% include path;"

and creates a segment consisting of the input segment and all segments included (see below for scanning procedure). The second entry checks the input segment to see if the first character is "%". If it is, expand_seg$percent proceeds as if it were expand_seg; otherwise it returns. Expand_seg always attempts to create an expanded segment; expand_seg$percent quits if the first character is not "%".

The expand command (BX.7.05) and the merge editor call expand_seg. The user also may call expand_seg but whenever feasible he should use the expand command (BX.7.05).

Usage

```
call expand_seg   /*or expand_seg$percent*/ (pathname,
                     mode, status);

dcl  pathname char(*)  /*or char (*) varying*/,

mode char (5)          /*or char (5) varying*/,

status fixed bin (17);
```

pathname is the path name in the file system hierarchy of the input segment that should be scanned.

mode is the access mode of the user which should be assigned to the output segment. If mode is a zero-length string or the null character string (""), expand_seg uses the mode assigned to the input file.

status is a number returned by expand_seg to indicate the result of the expansion. (See Appendix for specific values).

The "include" statement

The expand_seg procedure scans the input segment pathname for text in the form:

```
     % include path;
```

and replaces this piece of text by the segment located in the file system hierarchy at path.

The included segment need not be any particular syntactic unit: It may be a part of an expression or statement, a declaration, an internal procedure, etc. However, both the original segment and each included segment must be properly balanced with respect to both the comment convention and string quotes. Further, an included segment may not appear inside a comment or string.

"include" statements may be nested.

Interpretation of "path"

The expand_seg procedure evaluates path and converts it to an absolute path name when necessary. (An absolute path name is a path name relative to the root directory and begins with the character ">" or one of the special character strings enclosed by braces which represents an absolute path name - see below).

1. {root} represents the root directory.

2. {wdir} represents the absolute path name of the current working directory.

3. {pdir} represents the absolute path name of the current process directory.

4. {ldir} represents the absolute path name of the current system library directory.

5. {cdir} represents the absolute path name of the calling directory, i.e., the directory of the procedure segment which invoked expand_seg.

6. {=dir} represents the absolute pathname of the directory in which the segment containing "% include path;" resides.

7. {=path} represents the absolute pathname of the segment containing "% include path;"

8. {search_for callname} represents the absolute path name of the segment callname. (This is equivalent to searching for the pathname).

9. all other path names are interpreted relative to the current working directory.

Examples:

1. a>b

represents the path name "a>b" relative to the current working directory. It can also be represented by:

{wdir}>a>b

2. {ldir}_1>a

represents the segment having the entry name "a" which resides in the first system "sub-library" directory. (See BY.0.01 for a discussion of system libraries).

3.a. If the segment located at >c>d>x contains

"% include {=dir}>y;"

then

the segment to be included resides at

>c>d>y

b. If the input segment from 3a. contains

"% include {=path}.more;"

then

the segment to be included resides at

>c>d>x.more

Scanning Conventions

In this section the word "space" stands for any combination of the characters space, tab and newline.

Optional spaces may be used to separate the percent sign "%" from the include, from the pathname path, from the semi-colon ";".

expand_seg ignores the sequence

%;

expand_seg scans for comments and strings; hence, a percent sign occuring within either will be ignored. (Note: expand_seg assumes that the double quote (") delimits a string and "/*" - "*/" pairs delimit comments-thus an eplbsa programmer must be especially careful if he uses the include statement. At some later date expand_seg may be modified to recognize various delimiters and/or programming languages).

Implementation

expand_seg does the following:

1. calls entry_status (BY.2.10) to establish that the input segment exists as a non-directory branch (or non-directory link) and can be read by the user. If the preceding conditions do not hold then expand_seg sets the appropriate value for status and returns to its callers. (The possible values for status are described later in this section).

2. calls smm (BD.3.05) to obtain the pointer to the input segment.

2a. calls working_segs (BY.11.06) to obtain the pointer to a temporary working segment.

3. copies the input segment into the working segment until it encounters the string

"% include path;"

which is neither within a comment nor within a string. If it reaches the end of the input segment it goes to step 7.

4. evaluates path.

a. {wdir}, {ldir}, {pdir}, {cdir}, and {search_for callname} are evaluated by calling the respective routines (BY.17 and BX.13.04).

b. {root}, {=dir} and {=path} are evaluated in-line.

c. calls setpath (BY.2.04) to make pathname an absolute path name acceptable to the file system.

5. goes back to step 3 with path as the new input segment in order to expand the segment to be included in case the latter also wants segments to be included, i.e., nested "%include"'s.

6. calls smm to terminate the input segment.
(Steps 3-6 comprise a recursive internal procedure).

7. calls working_segs to move the working segment to the proper directory and delete the working segment.

a. If the entry pathname.expanded already exists, the working segment replaces it-either as a branch or at the end of a link.

b. If the entry pathname.expanded does not exist, the expanded segment is put in the working directory.

Initial Multics Version

A few restrictions have been adopted for the Initial Multics version of expand_seg:

1) Comments and strings are not scanned for and must not contain or be contained in include statements.

2) The "%" must immediately follow a new line character.

3) Only two forms of include statement are permitted:

```
%include path;
% include path;
```

4) No nesting of include statements is allowed (i.e., the user must take care of processing any nested includes himself).

5) Only absolute path names or path names relative to the working directory are permitted (i.e., the special character strings in path names are not permitted).