Published: 06/01/67
(Supersedes: BX.8.04, 12/07/66,
BX.8.04, 01/14/65)

Identification

Link
P. Smith, E. Bjorkman, R. J. Sobecki

Purpose

The link command provides a convenient way for the user to add link entries to any of his directories.

Usage

link (name1 pathname1 name2 pathname2...)

For each pair of arguments, namei and pathnamei, link establishes an entry in one of the user's directories. This entry is a link to another entry defined by the pathname, pathnamei. The entry defined by pathnamei may point to a non-directory segment, a directory, or another link. If namei does not specify a pathname relative to the root directory, the entry is established in a directory specified relative to the user's current working directory. If namei contains a pathname relative to the root directory, the entry is established in the directory specified by that pathname (i.e., the pathname contained in the string namei).

Comments

If namei is not unique in the directory the user is asked "Do you wish to delete namei?" When the answer is other than "yes", no action is taken on the i-th pair of arguments. If the answer is "yes", delete_entry (see BY.2.01) is called to delete the entry namei.

Implementation

link (name1 pathname1 name2 pathname2 ...)

Each link entry defined by namei and pathnamei is appended to a directory (defined implicitly or explicitly by namei) by a library routine, append_link (BY.2.01), which simply calls the Directory Supervisor primitive append1 (BY.8.02).

For each pair of arguments, namei and pathnamei, the following is done:

1) The pathname of the directory into which the link entry is to be appended is established by a call to a library routine, entryarg (BY.2.04). Entryarg calls one of two other library routines, wdir (BY.2.05) and setpath (BY.2.04), to obtain either the current working directory (entryarg calls wdir if namei does not contain a pathname relative to the root directory) or the specified pathname (entryarg calls setpath if namei does contain a pathname relative to the root directory) corresponding to the symbolic pathname specified in namei. Entryarg returns the pathname of the directory, from_pathname, and the entry name, from_entry, defined by namei. (Note: in a sense, the link is made from the directory, from_pathname, to an entry in another directory.)

2) entryarg is then called to construct the pathname to which the link is to be made, to_pathname, and an entry name, to_entry, from the symbolic pathname, pathnamei. This step is necessary because pathnamei may contain a pathname relative to the current working directory, whereas the link must be made to an entry relative to the root directory.

3) Upon return from the second call to entryarg, link concatenates to_pathname, the greater than symbol (">"), and to_entry. The result of the concatenation, link_to, is relative to the root directory.

4) The following call to append_link is made:

```
call append_link (from_pathname, from_entry, link_to);
```

5) Steps 1 through 4 above are executed for each pair of arguments. The next pair of arguments is checked to be sure there is an even number of arguments. After all the argument pairs have been processed, link returns.

The following is a list of possible errors, and for each error the corresponding long and short messages placed in the output stream, and the resultant action taken by the link command. Long messages are output if the brief option is off, short messages are output if the option is on.

1) a) Arguments are in an incorrect format for the link command (i.e. no arguments or an odd number of arguments).

b) short_message = "Link command incorrect."
long_message = "Link command incorrect. Correct format is:
link (name1 pathname1 name2 pathname2 ...)."

c) action: Link returns to its caller.

2) a) The append attribute is not on in the directory.

b) short_message = "No append attribute."
long_message = "Append attribute not on in the directory,"
||from_pathname||. "Links cannot be made"

c) action: This pair of arguments is left and the next pair of arguments is processed.

3) a) The directory is not found.

b) short_message = from_pathname|| "not found."
long_message = "directory, "||from_pathname||
"not found."

c) action: The next pair of arguments is processed.

4) a) There is no more space in the directory.

b) short_message = "No space in "||from_pathname||
" for "||from_entry
long_message = "Directory "||from_pathname||
" is out of space. Link "||
from_entry||" cannot be made."

c) action: The next pair of arguments is processed.

5) a) Some other error occurred in the call to append_link.

b) long_message,short_message = "Error "||error_code||"
occurred in append_link.
See BG.8.02."

c) action: The next pair of arguments is processed.

6) a) the name, from_entry, is not unique in the directory;

b) 1) short_message = from_entry||" not unique";
long_message = "Name "||from_entry||" not unique in directory "||from_pathname;

c) action:

1) If the no_questions option is on then this pair of arguments is ignored and link moves to the next pair.

2) If the no_questions option is off then the question is asked:

"Do you wish to delete old "||from_entry||"?";

a) If the answer is "no", this pair of arguments is ignored and link moves to the next pair.

b) If the answer is "yes" link calls delete_entry (BY.2.01) to delete the old entry. If the delete_entry call is successful, append_link is recalled.

c) If the answer is "yes" and the delete_entry call is not successful (i.e., the user does not have write access to the directory), the action taken is specified in errors 8-11 below, which are the errors for the call to delete_entry.

7) a) Delete_entry finds the write attribute not on in the directory or in the branch.

b) short_message = "Write attribute not on either in directory or in branch for " ||namei||". Entry not deleted";
long_message" = "Write attribute not on in either "||from_pathname|| "or in branch, "||from_entry||". Entry not deleted";

c) action: The next pair of arguments is processed.

8) a) The directory or entry name cannot be found by delete_entry.

b) long_message, short_message = from_pathname ||"or"||from_entry || "not found."

c) action: The next pair of arguments is processed.

9) a) Delete_entry finds that the entry points to a non-empty directory.

b) long_message, short_message = from_entry||" points to a non empty directory. Entry not deleted";

c) action: The next pair of arguments is processed.

10) a) Some other error occurred in delete_entry.

b) long_message, short_message = "Error "||error_code||" occurred in delete_entry. See BG.8.02";

c) action: The next pair of arguments is processed.