07/09/69
Edmeyer

Identification

LSM Utility Include File
lsminc.incl.epl

Edwin W. Meyer, Jr.

Purpose

lsminc comprises a number of internal procedures intended for use in manipulating LSM-format list structure.

Usage

The procedures of lsminc are declared as internal procedures within the include segment itself, and thus the point of inclusion must be before the use of any of these precedures. lsminc assumes the pointer 'sysp' to be the base pointer to the working lsm segment. 'sysp' must be declared prior to the inclusion of lsminc.

```
node_35 = mkfix (fix 35);
fix_35 = gtfix (node 35);

dcl fix_35 fixed bin (35), /*binary number*/
node_35 fixed bin (35); /*node of binary lsm data type*/
```

mkfix creates a binary lsm data type having the value 'fix_35' and returns its node in 'node_35,' while gtfix retrieves the value given the node.

```
node_35 = mkchar (char_string);
char_string = gtchar (node_35);
```

```
dcl node_35 fixed bin (35); /*node of character data block*/
char_string char (*); /*any type of character declaration
                        permissible in calling procedure*/
```

mkchar creates a character string block with the value 'char_string' and returns the block node in 'node 35'. gtchar retrieves the value of a character block, given its node.

```
node_35 = mkbit (bit_string);
bit_string = gtbit (node_35);

dcl node_35 fixed bin (35), /*node of bit string block*/
bit_string bit (*);
```

The operation of mkbit and gtbit is analogous to that of mkchar and gtchar.

```
array_node = mkarr (node_a, node_b, ..., node_n);
node = gtarr (array_node, array_index);
call setarr (array_node, array_index, node);

dcl array_node fixed bin (35),  /*node of array data block*/
(node_a, node_b, ..., node_n) fixed bin (35), /*nodes of
                        components of node array block*/
node fixed bin (35), /*a component node*/
array_index fixed bin (17); /*index to node array component-
                        lower bound is zero*/
```

mkarr creates a node array block using the supplied nodes as components, returning the node of the array in 'array_node.'

gtarr returns the 'array_index' th node of the array block at 'array_node.' The lower bound of 'array_index' is zero.

setarr writes 'node' into the 'array_index'th node of the array block at 'array_node.'

```
call print (node35);
        dcl node35 fixed bin(35);  /*data block node*/
```

print converts a character, bit, or the first number of a fixed array block to ascii and sends it to user_output (printed on-line). If 'node35' is of an illegal data type, no action is taken.

```
        parsed_node = parse (char_string);

        dcl parsed_node fixed bin(35), /* of returned node array*/
        char_string char (*); /*input character string*/
```

Parses char_string into several tokens whose delimiters are one or more blanks. Character blocks having the value of these tokens are created. The node of these blocks are placed into a created node array whose node is returned in 'parsed_node.' A null token ("" - double quotes)

generates a null node; no character string block is created to contain it.

```
bit1 = equal (node_a, node_b);

dcl bit1 bit(1), /*value of comparison*/
(node_a, node_b) fixed bin(35); /*nodes under comparison*/
```

bit1 is returned as "1"b if the two nodes are equivalent branches; otherwise bit1 is set to "0"b. Two hash tables of the same length and indirect types are equal by definition.