Identification

The EPL run-time routine, catstr_.
catstr_$catstrc_
catstr_$catstrb_

Ruth A. Jciss

Purpose

Catstr_ implements the PL/I string concatenation operations.

Usage

Catstr_ accepts any number of strings either varying or non-varying as arguments. The last argument is the result of concatenating the preceding arguments. If the last string is non-varying and has a greater length than the sum of the lengths of the concatenated strings, the concatenated string is extended on the right with a padding byte. The byte is 'O' b for catstrb_ and ASCII blank for catstrc_.

The calls are

\[
\begin{align*}
\text{call catstr$_n$catstrc$(s_1, s_2, \ldots, s_n, c)$} \\
\quad c = s_1 \| s_2 \| \ldots \| s_n \\
\text{call catstr$_n$catstrb$(s_1, s_2, \ldots, s_n, b)$} \\
\quad b = s_1 \| s_2 \| \ldots \| s_n \\
\end{align*}
\]

where sj is a string, c is a character string and b is a bit string.

EPL has not been implemented to concatenate more than two strings at a time but by using the call statement an EPL programmer can prevent the compiler from creating unnecessary temporary storage. If the last string is non-varying and any of the concatenated strings overlaps it, a varying string temporary is created for the answer which is moved to the proper non-varying location when the concatenation is completed. This avoids trouble with a statement such as

\[ b = a \| b \]
EFL compiles either of two calls to stgop_ to concatenate strings. stgop_ will then always call catstr_ as follows:

```plaintext
call stgop_$ctcs_(c1,c2,c3);
call catstr_$catstrc_(c1,c2,c3);
c3=c1||c2;
call stgop_$ctbs_(b1,b2,b3);
call catstr_$catstrb_(b1,b2,b3);
b3=b1||b2;
```

where c1, c2, and c3 are character strings and b1, b2, and b3 are bit strings.

**Error**

If any argument is not a string, will stop on oct 0.