NAME

Tcl_DStringInit, Tcl_DStringAppend, Tcl_DStringAppendElement, Tcl_DStringStartSublist, Tcl_DStringEndSublist, Tcl_DStringLength, Tcl_DStringValue, Tcl_DStringSetLength, Tcl_DStringTrunc, Tcl_DStringFree, Tcl_DStringResult, Tcl_DStringGetResult − manipulate dynamic strings

SYNOPSIS

#include <tcl.h>

Tcl_DStringInit(*dsPtr*)

char *
Tcl_DStringAppend(*dsPtr, bytes, length*)

char *
Tcl_DStringAppendElement(*dsPtr, element*)

Tcl_DStringStartSublist(*dsPtr*)

Tcl_DStringEndSublist(*dsPtr*)

int
Tcl_DStringLength(*dsPtr*)

char *
Tcl_DStringValue(*dsPtr*)

Tcl_DStringSetLength(*dsPtr, newLength*)

Tcl_DStringTrunc(*dsPtr, newLength*)

Tcl_DStringFree(*dsPtr*)

Tcl_DStringResult(*interp, dsPtr*)

Tcl_DStringGetResult(*interp, dsPtr*)

ARGUMENTS

Tcl_DString **dsPtr* (in/out)	Pointer to structure that is used to manage a dynamic string.
const char **bytes* (in)	Pointer to characters to append to dynamic string.
const char **element* (in)	Pointer to characters to append as list element to dynamic string.
int *length* (in)	Number of bytes from *bytes* to add to dynamic string. If -1, add all characters up to null terminating character.
int *newLength* (in)	New length for dynamic string, not including null terminating character.
Tcl_Interp ***interp* (in/out)	Interpreter whose result is to be set from or moved to the dynamic string.

DESCRIPTION

Dynamic strings provide a mechanism for building up arbitrarily long strings by gradually appending information. If the dynamic string is short then there will be no memory allocation overhead; as the string gets

larger, additional space will be allocated as needed.

Tcl_DStringInit initializes a dynamic string to zero length. The Tcl_DString structure must have been allocated by the caller. No assumptions are made about the current state of the structure; anything already in it is discarded. If the structure has been used previously, **Tcl_DStringFree** should be called first to free up any memory allocated for the old string.

Tcl_DStringAppend adds new information to a dynamic string, allocating more memory for the string if needed. If*length* is less than zero then e verything in *bytes* is appended to the dynamic string; otherwise *length* specifies the number of bytes to append. **Tcl_DStringAppend** returns a pointer to the characters of the new string. The string can also be retrieved from the *string* field of the Tcl_DString structure.

Tcl_DStringAppendElement is similar to **Tcl_DStringAppend** except that it does not take a *length* argument (it appends all of *element*) and it converts the string to a proper list element before appending. **Tcl_DStringAppendElement** adds a separator space before the new list element unless the new list element is the first in a list or sub-list (i.e. either the current string is empty, or it contains the single character "{", or the last two characters of the current string are " {"). **Tcl_DStringA ppendElement** returns a pointer to the characters of the new string.

Tcl_DStringStartSublist and **Tcl_DStringEndSublist** can be used to create nested lists. To append a list element that is itself a sublist, first call **Tcl_DStringStartSublist**, then call **Tcl_DStringAppendElement** for each of the elements in the sublist, then call **Tcl_DStringEndSublist** to end the sublist. **Tcl_DStringStartSublist** appends a space character if needed, followed by an open brace; **Tcl_DStringEndSublist** appends a close brace. Lists can be nested to any depth.

Tcl_DStringLength is a macro that returns the current length of a dynamic string (not including the terminating null character). **Tcl_DStringV alue** is a macro that returns a pointer to the current contents of a dynamic string.

Tcl_DStringSetLength changes the length of a dynamic string. If *newLength* is less than the string's current length, then the string is truncated. If *newLength* is greater than the string's current length, then the string will become longer and new space will be allocated for the string if needed. However, **Tcl_DStringSetLength** will not initialize the new space except to provide a terminating null character; it is up to the caller to fill in the new space. **Tcl_DStringSetLength** does not free up the string' s storage space even if the string is truncated to zero length, so **Tcl_DStringFree** will still need to be called.

Tcl_DStringTrunc changes the length of a dynamic string. This procedure is now deprecated. **Tcl_DStringSetLength** should be used instead.

Tcl_DStringFree should be called when you are finished using the string. It frees up any memory that was allocated for the string and reinitializes the string's value to an empty string.

Tcl_DStringResult sets the result of *interp* to the value of the dynamic string given by *dsPtr*. It does this by moving a pointer from *dsPtr* to the interpreter's result. This saves the cost of allocating new memory and copying the string. **Tcl_DStringResult** also reinitializes the dynamic string to an empty string.

Tcl_DStringGetResult does the opposite of **Tcl_DStringResult**. It sets the value of *dsPtr* to the result of *interp* and it clears *interp*'s result. If possible it does this by moving a pointer rather than by copying the string.

KEYWORDS

append, dynamic string, free, result