NAME

Tcl_CreateCommand, Tcl_DeleteCommand − define application-specific command bindings

SYNOPSIS

#include <tcl.h>

Tcl_CreateCommand(*interp, cmdName, proc, clientData, deleteProc*)

int
Tcl_DeleteCommand(*interp, cmdName*)

ARGUMENTS

Tcl_Interp	**interp*	(in)	Interpreter in which to create new command.
char	**cmdName*	(in)	Name of command to create or delete.
Tcl_CmdProc	**proc*	(in)	Implementation of new command: *proc* will be called whenever *cmdName* is invoked as a command.
ClientData	*clientData*	(in)	Arbitrary one-word value to pass to *proc* and *deleteProc*.
Tcl_CmdDeleteProc	**deleteProc*	(in)	Procedure to call before *cmdName* is deleted from the interpreter; allows for command-specific cleanup. If NULL, then no procedure is called before the command is deleted.

DESCRIPTION

Tcl_CreateCommand defines a new command in *interp* and associates it with procedure *proc* such that whenever *cmdName* is invoked as a Tcl command (via a call to **Tcl_Eval**) the Tcl interpreter will call *proc* to process the command. If there is already a command *cmdName* associated with the interpreter, it is deleted. *Proc* should have arguments and result that match the type **Tcl_CmdProc**:

```
typedef int Tcl_CmdProc(
        ClientData clientData,
        Tcl_Interp *interp,
        int argc,
        char *argv[]);
```

When *proc* is invoked the *clientData* and *interp* parameters will be copies of the *clientData* and *interp* arguments given to **Tcl_CreateCommand**. Typically, *clientData* points to an application-specific data structure that describes what to do when the command procedure is invoked. *Argc* and *argv* describe the arguments to the command, *argc* giving the number of arguments (including the command name) and *argv* giving the values of the arguments as strings. The *argv* array will contain *argc*+1 values; the first *argc* values point to the argument strings, and the last value is NULL.

Proc must return an integer code that is either **TCL_OK**, **TCL_ERROR**, **TCL_RETURN**, **TCL_BREAK**, or **TCL_CONTINUE**. See the Tcl overview man page for details on what these codes mean. Most normal commands will only return **TCL_OK** or **TCL_ERROR**. In addition, *proc* must set *interp->result* to point to a string value; in the case of a **TCL_OK** return code this gives the result of the command, and in the case of **TCL_ERROR** it gives an error message. The **Tcl_SetResult** procedure provides an easy interface for setting the return value; for complete details on how the *interp->result* field is managed, see the **Tcl_Interp** man page. Before invoking a command procedure, **Tcl_Eval** sets *interp->result* to point to an empty string, so simple commands can return an empty result by doing nothing at all.

The contents of the *argv* array are copies made by the Tcl interpreter for the use of *proc*. *Proc* may alter any of the strings in *argv*. However, the *argv* array is recycled as soon as *proc* returns, so *proc* must not set *interp->result* to point anywhere within the *argv* values (call Tcl_SetResult with status **TCL_VOLATILE** if you want to return something from the *argv* array).

DeleteProc will be invoked when (if) *cmdName* is deleted. This can occur through a call to **Tcl_DeleteCommand** or **Tcl_DeleteInterp**, or by replacing *cmdName* in another call to Tcl_CreateCommand. *DeleteProc* is invoked before the command is deleted, and gives the application an opportunity to release any structures associated with the command. *DeleteProc* should have arguments and result that match the type **Tcl_CmdDeleteProc**:

```
typedef void Tcl_CmdDeleteProc(ClientData clientData);
```

The *clientData* argument will be the same as the *clientData* argument passed to **Tcl_CreateCommand**.

Tcl_DeleteCommand deletes a command from a command interpreter. Once the call completes, attempts to invoke *cmdName* in *interp* will result in errors. If *cmdName* isn't bound as a command in *interp* then **Tcl_DeleteCommand** does nothing and returns -1; otherwise it returns 0. There are no restrictions on *cmdName*: it may refer to a built-in command, an application-specific command, or a Tcl procedure.

KEYWORDS

bind, command, create, delete, interpreter