NAME

Tcl_CreateTrace, Tcl_DeleteTrace − arrange for command execution to be traced

SYNOPSIS

#include <tcl.h>

Tcl_Trace
Tcl_CreateTrace(*interp, level, proc, clientData*)

Tcl_DeleteTrace(*interp, trace*)

ARGUMENTS

Tcl_Interp	**interp*	(in)	Interpreter containing command to be traced or untraced.
int	*level*	(in)	Only commands at or below this nesting level will be traced. 1 means top-level commands only, 2 means top-level commands or those that are invoked as immediate consequences of executing top-level commands (procedure bodies, bracketed commands, etc.) and so on.
Tcl_CmdTraceProc	**proc*	(in)	Procedure to call for each command that's executed. See below for details on the calling sequence.
ClientData	*clientData*	(in)	Arbitrary one-word value to pass to *proc*.
Tcl_Trace	*trace*	(in)	Token for trace to be removed (return value from previous call to **Tcl_CreateTrace**).

DESCRIPTION

Tcl_CreateTrace arranges for command tracing. From now on, *proc* will be invoked before Tcl calls command procedures to process commands in *interp*. The return value from **Tcl_CreateTrace** is a token for the trace, which may be passed to **Tcl_DeleteTrace** to remove the trace. There may be many traces in effect simultaneously for the same command interpreter.

Proc should have arguments and result that match the type **Tcl_CmdTraceProc**:

```
typedef void Tcl_CmdTraceProc(
        ClientData clientData,
        Tcl_Interp *interp,
        int level,
        char *command,
        Tcl_CmdProc *cmdProc,
        ClientData cmdClientData,
        int argc,
        char *argv[]));
```

The *clientData* and *interp* parameters are copies of the corresponding arguments given to **Tcl_CreateTrace**. *ClientData* typically points to an application-specific data structure that describes what to do when *proc* is invoked. *Level* gives the nesting level of the command (1 for top-level commands passed to **Tcl_Eval** by the application, 2 for the next-level commands passed to **Tcl_Eval** as part of parsing or interpreting level-1 commands, and so on). *Command* points to a string containing the text of the command, before any argument substitution. *CmdProc* contains the address of the command procedure that will be called to process the command (i.e. the *proc* argument of some previous call to **Tcl_CreateCommand**) and *cmdClientData* contains the associated client data for *cmdProc* (the *clientData* value passed to **Tcl_CreateCommand**). *Argc* and *argv* give the final argument information that will be passed to *cmdProc*, after

command, variable, and backslash substitution. *Proc* must not modify the *command* or *argv* strings.

Tracing will only occur for commands at nesting level less than or equal to the *level* parameter (i.e. the *level* parameter to *proc* will always be less than or equal to the *level* parameter to **Tcl_CreateTrace**).

Calls to *proc* will be made by the Tcl parser immediately before it calls the command procedure for the command (*cmdProc*). This occurs after argument parsing and substitution, so tracing for substituted commands occurs before tracing of the commands containing the substitutions. If there is a syntax error in a command, or if there is no command procedure associated with a command name, then no tracing will occur for that command. If a string passed to Tcl_Eval contains multiple commands (bracketed, or on different lines) then multiple calls to *proc* will occur, one for each command. The *command* string for each of these trace calls will reflect only a single command, not the entire string passed to Tcl_Eval.

Tcl_DeleteTrace removes a trace, so that no future calls will be made to the procedure associated with the trace. After **Tcl_DeleteTrace** returns, the caller should never again use the *trace* token.

KEYWORDS

command, create, delete, interpreter, trace