NAME

Tcl_Eval, Tcl_VarEval, Tcl_EvalFile, Tcl_GlobalEval − execute Tcl commands

SYNOPSIS

#include <tcl.h>

int
Tcl_Eval(*interp, cmd, flags, termPtr*)

int
Tcl_VarEval(*interp, string, string, ...* **(char *) NULL**)

int
Tcl_EvalFile(*interp, fileName*)

int
Tcl_GlobalEval(*interp, cmd*)

ARGUMENTS

Tcl_Interp	**interp*	(in)	Interpreter in which to execute the command. String result will be stored in *interp->result*.
char	**cmd*	(in)	Command (or sequence of commands) to execute. Must be in writable memory (Tcl_Eval makes temporary modifications to the command).
int	*flags*	(in)	Either **TCL_BRACKET_TERM** or 0. If 0, then **Tcl_Eval** will process commands from *cmd* until it reaches the null character at the end of the string. If **TCL_BRACKET_TERM**, then **Tcl_Eval** will process comands from *cmd* until either it reaches a null character or it encounters a close bracket that isn’t backslashed or enclosed in braces, at which point it will return. Under normal conditions, *flags* should be 0.
char	***termPtr*	(out)	If *termPtr* is non-NULL, **Tcl_Eval** fills in **termPtr* with the address of the character just after the last one in the last command successfully executed (normally the null character at the end of *cmd*). If an error occurs in the first command in *cmd*, then **termPtr* will be set to *cmd*.
char	**string*	(in)	String forming part of Tcl command.
char	**fileName*	(in)	Name of file containing Tcl command string.

DESCRIPTION

All four of these procedures execute Tcl commands. **Tcl_Eval** is the core procedure: it parses commands from *cmd* and executes them in order until either an error occurs or **Tcl_Eval** reaches a terminating character (’]’ or ’\0’, depending on the value of *flags*). The return value from **Tcl_Eval** is one of the Tcl return codes **TCL_OK**, **TCL_ERROR**, **TCL_RETURN**, **TCL_BREAK**, or **TCL_CONTINUE**, and *interp->result* will point to a string with additional information (result value or error message). This return information corresponds to the last command executed from *cmd*.

Tcl_VarEval takes any number of string arguments of any length, concatenates them into a single string, then calls **Tcl_Eval** to execute that string as a Tcl command. It returns the result of the command and also modifies *interp->result* in the usual fashion for Tcl commands. The last argument to **Tcl_VarEval** must be NULL to indicate the end of arguments.

Tcl_EvalFile reads the file given by *fileName* and evaluates its contents as a Tcl command by calling **Tcl_Eval**. It returns a standard Tcl result that reflects the result of evaluating the file. If the file couldn't be read then a Tcl error is returned to describe why the file couldn't be read.

Tcl_GlobalEval is similar to **Tcl_Eval** except that it processes the command at global level. This means that the variable context for the command consists of global variables only (it ignores any Tcl procedure that is active). This produces an effect similar to the Tcl command ''**uplevel 0**''.

During the processing of a Tcl command it is legal to make nested calls to evaluate other commands (this is how conditionals, loops, and procedures are implemented). If a code other than **TCL_OK** is returned from a nested **Tcl_Eval** invocation, then the caller should normally return immediately, passing that same return code back to its caller, and so on until the top-level application is reached. A few commands, like **for**, will check for certain return codes, like **TCL_BREAK** and **TCL_CONTINUE**, and process them specially without returning.

Tcl_Eval keeps track of how many nested Tcl_Eval invocations are in progress for *interp*. If a code of **TCL_RETURN**, **TCL_BREAK**, or **TCL_CONTINUE** is about to be returned from the topmost **Tcl_Eval** invocation for *interp*, then **Tcl_Eval** converts the return code to **TCL_ERROR** and sets *interp->result* to point to an error message indicating that the **return**, **break**, or **continue** command was invoked in an inappropriate place. This means that top-level applications should never see a return code from **Tcl_Eval** other then **TCL_OK** or **TCL_ERROR**.

KEYWORDS

command, execute, file, global, interpreter, variable