NAME

Tcl_AddErrorInfo, Tcl_SetErrorCode, Tcl_UnixError, Tcl_CheckStatus − record information about errors

SYNOPSIS

#include <tcl.h>

char *
Tcl_AddErrorInfo(*interp, message*)

void
Tcl_SetErrorCode(*interp, element, element, ...*)

char *
Tcl_UnixError(*interp*)

ARGUMENTS

Tcl_Interp	**interp*	(in)	Interpreter in which to record information.
char	**message*	(in)	Identifying string to record in **errorInfo** variable.
char	**element*	(in)	String to record as one element of **errorCode** variable. Last *element* argument must be NULL.

DESCRIPTION

These procedures are used to manipulate two global variables that hold information about errors. The variable **errorInfo** holds a stack trace of the operations that were in progress when an error occurred, and is intended to be human-readable. The variable **errorCode** holds a list of items that are intended to be machine-readable. The first item in **errorCode** identifies the class of error that occurred (e.g. UNIX means an error occurred in a Unix system call) and additional elements in **errorCode** hold additional pieces of information that depend on the class. See the Tcl overview manual entry for details on the various formats for **errorCode**.

The **errorInfo** variable is gradually built up as an error unwinds through the nested operations. Each time an error code is returned to **Tcl_Eval** it calls the procedure **Tcl_AddErrorInfo** to add additional text to **errorInfo** describing the command that was being executed when the error occurred. By the time the error has been passed all the way back to the application, it will contain a complete trace of the activity in progress when the error occurred.

It is sometimes useful to add additional information to **errorInfo** beyond what can be supplied automatically by **Tcl_Eval**. **Tcl_AddErrorInfo** may be used for this purpose: its *message* argument contains an additional string to be appended to **errorInfo**. For example, the **source** command calls **Tcl_AddErrorInfo** to record the name of the file being processed and the line number on which the error occurred; for Tcl procedures, the procedure name and line number within the procedure are recorded, and so on. The best time to call **Tcl_AddErrorInfo** is just after **Tcl_Eval** has returned **TCL_ERROR**. In calling **Tcl_AddErrorInfo**, you may find it useful to use the **errorLine** field of the interpreter (see the **Tcl_Interp** manual entry for details).

The procedure **Tcl_SetErrorCode** is used to set the **errorCode** variable. Its *element* arguments give one or more strings to record in **errorCode**: each *element* will become one item of a properly-formed Tcl list stored in **errorCode**. **Tcl_SetErrorCode** is typically invoked just before returning an error. If an error is returned without calling **Tcl_SetErrorCode** then the Tcl interpreter automatically sets **errorCode** to **NONE**.

Tcl_UnixError sets the **errorCode** variable after an error in a UNIX kernel call. It reads the value of the **errno** C variable and calls **Tcl_SetErrorCode** to set **errorCode** in the **UNIX** format. In addition, **Tcl_UnixError** returns a human-readable diagnostic message for the error (this is the same value that will

appear as the third element in **errorCode**). It may be convenient to include this string as part of the error message returned to the application in *interp->result*.

It is important to call the procedures described here rather than setting **errorInfo** or **errorCode** directly with **Tcl_SetVar**. The reason for this is that the Tcl interpreter keeps information about whether these procedures have been called. For example, the first time **Tcl_AppendResult** is called for an error, it clears the existing value of **errorInfo** and adds the error message in *interp->result* to the variable before appending *message*; in subsequent calls, it just appends the new *message*. When **Tcl_SetErrorCode** is called, it sets a flag indicating that **errorCode** has been set; this allows the Tcl interpreter to set **errorCode** to **NONE if it receives an error return when Tcl_SetErrorCode** hasn't been called.

If the procedure **Tcl_ResetResult** is called, it clears all of the state associated with **errorInfo** and **errorCode** (but it doesn't actually modify the variables). If an error had occurred, this will clear the error state to make it appear as if no error had occurred after all.

SEE ALSO

Tcl_ResetResult, Tcl_Interp

KEYWORDS

error, stack, trace, variable