NAME

Tcl_Interp − client-visible fields of interpreter structures

SYNOPSIS

```
#include <tcl.h>

typedef struct {
        char *result;
        Tcl_FreeProc *freeProc;
        int errorLine;
} Tcl_Interp;

typedef void Tcl_FreeProc(char *blockPtr);
```

DESCRIPTION

The **Tcl_CreateInterp** procedure returns a pointer to a Tcl_Interp structure. This pointer is then passed into other Tcl procedures to process commands in the interpreter and perform other operations on the interpreter. Interpreter structures contain many many fields that are used by Tcl, but only three that may be accessed by clients: *result*, *freeProc*, and *errorLine*.

The *result* and *freeProc* fields are used to return results or error messages from commands. This information is returned by command procedures back to **Tcl_Eval**, and by **Tcl_Eval** back to its callers. The *result* field points to the string that represents the result or error message, and the *freeProc* field tells how to dispose of the storage for the string when it isn't needed anymore. The easiest way for command procedures to manipulate these fields is to call procedures like **Tcl_SetResult** or **Tcl_AppendResult**; they will hide all the details of managing the fields. The description below is for those procedures that manipulate the fields directly.

Whenever a command procedure returns, it must ensure that the *result* field of its interpreter points to the string being returned by the command. The *result* field must always point to a valid string. If a command wishes to return no result then *interp->result* should point to an empty string. Normally, results are assumed to be statically allocated, which means that the contents will not change before the next time **Tcl_Eval** is called or some other command procedure is invoked. In this case, the *freeProc* field must be zero. Alternatively, a command procedure may dynamically allocate its return value (e.g. using **malloc**) and store a pointer to it in *interp->result*. In this case, the command procedure must also set *interp->freeProc* to the address of a procedure that can free the value (usually **free**). If *interp->freeProc* is non-zero, then Tcl will call *freeProc* to free the space pointed to by *interp->result* before it invokes the next command. If a client procedure overwrites *interp->result* when *interp->freeProc* is non-zero, then it is responsible for calling *freeProc* to free the old *interp->result* (the **Tcl_FreeResult** macro should be used for this purpose).

FreeProc should have arguments and result that match the **Tcl_FreeProc** declaration above: it receives a single argument which is a pointer to the result value to free. In most applications **free** is the only non-zero value ever used for *freeProc*. However, an application may store a different procedure address in *freeProc* in order to use an alternate memory allocator or in order to do other cleanup when the result memory is freed.

As part of processing each command, **Tcl_Eval** initializes *interp->result* and *interp->freeProc* just before calling the command procedure for the command. The *freeProc* field will be initialized to zero, and *interp->result* will point to an empty string. Commands that do not return any value can simply leave the fields alone. Furthermore, the empty string pointed to by *result* is actually part of an array of **TCL_RESULT_SIZE** characters (approximately 200). If a command wishes to return a short string, it can simply copy it to the area pointed to by *interp->result*. Or, it can use the sprintf procedure to generate a short result string at the location pointed to by *interp->result*.

It is a general convention in Tcl-based applications that the result of an interpreter is normally in the initialized state described in the previous paragraph. Procedures that manipulate an interpreter's result (e.g. by returning an error) will generally assume that the result has been initialized when the procedure is called. If such a procedure is to be called after the result has been changed, then **Tcl_ResetResult** should be called first to reset the result to its initialized state.

The *errorLine* field is valid only after **Tcl_Eval** returns a **TCL_ERROR** return code. In this situation the *errorLine* field identifies the line number of the command being executed when the error occurred. The line numbers are relative to the command being executed: 1 means the first line of the command passed to **Tcl_Eval**, 2 means the second line, and so on. The *errorLine* field is typically used in conjunction with **Tcl_AddErrorInfo** to report information about where an error occurred. *ErrorLine* should not normally be modified except by **Tcl_Eval**.

KEYWORDS

free, initialized, interpreter, malloc, result