NAME

auto_execok, auto_import, auto_load, auto_mkindex, auto_qualify, auto_reset, tcl_findLibrary, parray, tcl_endOfWord, tcl_startOfNextWord, tcl_startOfPreviousWord, tcl_wordBreakAfter, tcl_wordBreakBefore − standard library of Tcl procedures

SYNOPSIS

auto_execok *cmd*
auto_import *pattern*
auto_load *cmd*
auto_mkindex *dir pattern pattern ...*
auto_qualify *command namespace*
auto_reset
tcl_findLibrary *basename version patch initScript enVarName varName*
parray *arrayName* ?*pattern*?
tcl_endOfWord *str start*
tcl_startOfNextWord *str start*
tcl_startOfPreviousWord *str start*
tcl_wordBreakAfter *str start*
tcl_wordBreakBefore *str start*

INTRODUCTION

Tcl includes a library of Tcl procedures for commonly-needed functions. The procedures defined in the Tcl library are generic ones suitable for use by many different applications. The location of the Tcl library is returned by the **info library** command. In addition to the Tcl library, each application will normally have its own library of support procedures as well; the location of this library is normally given by the value of the **$***app***_library** global variable, where *app* is the name of the application. For example, the location of the Tk library is kept in the variable **tk_library**.

To access the procedures in the Tcl library, an application should source the file **init.tcl** in the library, for example with the Tcl command

source [file join [info library] init.tcl]

If the library procedure **Tcl_Init** is invoked from an application's **Tcl_AppInit** procedure, this happens automatically. The code in**init.tcl** will define the **unkno wn** procedure and arrange for the other procedures to be loaded on-demand using the auto-load mechanism defined below.

COMMAND PROCEDURES

The following procedures are provided in the Tcl library:

auto_execok *cmd*

Determines whether there is an executable file or shell builtin by the name *cmd*. If so, it returns a list of arguments to be passed to **exec** to execute the executable file or shell builtin named by *cmd*. If not, it returns an empty string. This command examines the directories in the current search path (given by the PATH environment variable) in its search for an executable file named *cmd*. On Windows platforms, the search is expanded with the same directories and file extensions as used by **exec**. **Auto_execok** remembers information about previous searches in an array named **auto_execs**; this avoids the path search in future calls for the same *cmd*. The command**auto_r e-set** may be used to force **auto_execok** to forget its cached information.

auto_import *pattern*

Auto_import is invoked during **namespace import** to see if the imported commands specified by *pattern* reside in an autoloaded library. If so, the commands are loaded so that they will be available to the interpreter for creating the import links. If the commands do not reside in an autoloaded library, **auto_import** does nothing. The pattern matching is performed according to the matching rules of **namespace import**.

auto_load *cmd*

This command attempts to load the definition for a Tcl command named *cmd*. To do this, it searches an *auto-load path*, which is a list of one or more directories. The auto-load path is given by the global variable **auto_path** if it exists. If there is no**auto_path** v ariable, then the TCLLIB-PATH environment variable is used, if it exists. Otherwise the auto-load path consists of just the Tcl library directory. Within each directory in the auto-load path there must be a file **tclIndex** that describes one or more commands defined in that directory and a script to evaluate to load each of the commands. The **tclIndex** file should be generated with the **auto_mkindex** command. If *cmd* is found in an index file, then the appropriate script is evaluated to create the command. The **auto_load** command returns 1 if *cmd* was successfully created. The command returns 0 if there was no index entry for *cmd* or if the script did not actually define *cmd* (e.g. because index information is out of date). If an error occurs while processing the script, then that error is returned. **Auto_load** only reads the index information once and saves it in the array **auto_index**; future calls to **auto_load** check for *cmd* in the array rather than re-reading the index files. The cached index information may be deleted with the command **auto_reset**. This will force the next **auto_load** command to reload the index database from disk.

auto_mkindex *dir pattern pattern ...*

Generates an index suitable for use by **auto_load**. The command searches*dir* for all files whose names match any of the *pattern* arguments (matching is done with the **glob** command), generates an index of all the Tcl command procedures defined in all the matching files, and stores the index information in a file named **tclIndex** in *dir*. If no pattern is given a pattern of ***.tcl** will be assumed. For example, the command

auto_mkindex foo *.tcl

will read all the **.tcl** files in subdirectory **foo** and generate a new index file **foo/tclIndex**.

Auto_mkindex parses the Tcl scripts by sourcing them into a child interpreter and monitoring the proc and namespace commands that are executed. Extensions can use the (undocumented) auto_mkindex_parser package to register other commands that can contribute to the auto_load index. You will have to read through auto.tcl to see how this works.

Auto_mkindex_old (which has the same syntax as **auto_mkindex**) parses the Tcl scripts in a relatively unsophisticated way: if any line contains the word "**proc**" as its first characters then it is assumed to be a procedure definition and the next word of the line is taken as the procedure's name. Procedure definitions that do not appear in this way (e.g. they have spaces before the **proc**) will not be indexed. If your script contains "dangerous" code, such as global initialization code or procedure names with special characters like **$**, *****, **[** or **]**, you are safer using **auto_mkindex_old**.

auto_reset

Destroys all the information cached by **auto_execok** and **auto_load**. This information will be reread from disk the next time it is needed. **A uto_reset** also deletes any procedures listed in the auto-load index, so that fresh copies of them will be loaded the next time that they are used.

auto_qualify *command namespace*

Computes a list of fully qualified names for *command*. This list mirrors the path a standard Tcl interpreter follows for command lookups: first it looks for the command in the current namespace, and then in the global namespace. Accordingly, if *command* is relative and *namespace* is not **::**, the list returned has two elements: *command* scoped by *namespace*, as if it were a command in the *namespace* namespace; and *command* as if it were a command in the global namespace. Otherwise, if either *command* is absolute (it begins with **::**), or *namespace* is **::**, the list contains only *command* as if it were a command in the global namespace.

Auto_qualify is used by the auto-loading facilities in Tcl, both for producing auto-loading indexes such as *pkgIndex.tcl*, and for performing the actual auto-loading of functions at runtime.

tcl_findLibrary *basename version patch initScript enVarName varName*

This is a standard search procedure for use by extensions during their initialization. They call this procedure to look for their script library in several standard directories. The last component of the name of the library directory is normally *basenameversion* (e.g., tk8.0), but it might be "library" when in the build hierarchies. The *initScript* file will be sourced into the interpreter once it is found. The directory in which this file is found is stored into the global variable *varName*. If this variable is already defined (e.g., by C code during application initialization) then no searching is done. Otherwise the search looks in these directories: the directory named by the environment variable *enVarName*; relative to the Tcl library directory; relative to the executable file in the standard installation bin or bin/*arch* directory; relative to the executable file in the current build tree; relative to the executable file in a parallel build tree.

parray *arrayName* ?*pattern*?

Prints on standard output the names and values of all the elements in the array *arrayName*, or just the names that match *pattern* (using the matching rules of **string match**) and their values if *pattern* is given. *ArrayName* must be an array accessible to the caller of **parray**. It may be either local or global.

WORD BOUNDARY HELPERS

These procedures are mainly used internally by Tk.

tcl_endOfWord *str start*

Returns the index of the first end-of-word location that occurs after a starting index *start* in the string *str*. An end-of-word location is defined to be the first non-word character following the first word character after the starting point. Returns -1 if there are no more end-of-word locations after the starting point. See the description of **tcl_wordchars** and **tcl_nonwordchars** below for more details on how Tcl determines which characters are word characters.

tcl_startOfNextWord *str start*

Returns the index of the first start-of-word location that occurs after a starting index *start* in the string *str*. A start-of-word location is defined to be the first word character following a non-word character. Returns −1 if there are no more start-of-word locations after the starting point.

tcl_startOfPreviousWord *str start*

Returns the index of the first start-of-word location that occurs before a starting index *start* in the string *str*. Returns −1 if there are no more start-of-word locations before the starting point.

tcl_wordBreakAfter *str start*

Returns the index of the first word boundary after the starting index *start* in the string *str*. Returns −1 if there are no more boundaries after the starting point in the given string. The index returned refers to the second character of the pair that comprises a boundary.

tcl_wordBreakBefore *str start*

Returns the index of the first word boundary before the starting index *start* in the string *str*. Returns −1 if there are no more boundaries before the starting point in the given string. The index returned refers to the second character of the pair that comprises a boundary.

VARIABLES

The following global variables are defined or used by the procedures in the Tcl library. They fall into two broad classes, handling unknown commands and packages, and determining what are words.

AUTOLOADING AND PACKAGE MANAGEMENT VARIABLES

auto_execs

Used by **auto_execok** to record information about whether particular commands exist as executable files.

auto_index

Used by **auto_load** to save the index information read from disk.

auto_noexec
If set to any value, then **unknown** will not attempt to auto-exec any commands.

auto_noload
If set to any value, then **unknown** will not attempt to auto-load any commands.

auto_path
If set, then it must contain a valid Tcl list giving directories to search during auto-load operations (including for package index files when using the default **package unknown** handler). This variable is initialized during startup to contain, in order: the directories listed in the **TCLLIBPATH** environment variable, the directory named by the **tcl_library** global variable, the parent directory of **tcl_library**, the directories listed in the **tcl_pkgPath** variable. Additional locations to look for files and package indices should normally be added to this variable using **lappend**.

env(TCL_LIBRARY)
If set, then it specifies the location of the directory containing library scripts (the value of this variable will be assigned to the **tcl_library** variable and therefore returned by the command **info library**). If this variable is not set then a default value is used.

env(TCLLIBPATH)
If set, then it must contain a valid Tcl list giving directories to search during auto-load operations. Directories must be specified in Tcl format, using "/" as the path separator, regardless of platform. This variable is only used when initializing the **auto_path** variable.

WORD BOUNDARY DETERMINATION VARIABLES

These variables are only used in the **tcl_endOfWord**, **tcl_startOfNextWord**, **tcl_startOfPreviousWord**, **tcl_wordBreakAfter**, and **tcl_wordBreakBefore** commands.

tcl_nonwordchars
This variable contains a regular expression that is used by routines like **tcl_endOfWord** to identify whether a character is part of a word or not. If the pattern matches a character, the character is considered to be a non-word character. On Windows platforms, spaces, tabs, and newlines are considered non-word characters. Under Unix, everything but numbers, letters and underscores are considered non-word characters.

tcl_wordchars
This variable contains a regular expression that is used by routines like **tcl_endOfWord** to identify whether a character is part of a word or not. If the pattern matches a character, the character is considered to be a word character. On Windows platforms, words are comprised of any character that is not a space, tab, or newline. Under Unix, words are comprised of numbers, letters or underscores.

SEE ALSO

env(n), info(n), re_syntax(n)

KEYWORDS

auto-exec, auto-load, library, unknown, word, whitespace