

Tcl/Tk



Reference Guide

for Tcl 8.4.3 / Tk 8.4.3

Tk/Tcl program designed and created by
John Ousterhout
<<http://home.pacbell.net/ouster/>>

Reference guide contents written by
Paul Raines **<raines@slac.stanford.edu>**
Jeff Tranter **<tranter@pobox.com>**
Dave Bodensab **<dave@bodensab.org>**

Reference guide format designed and created by
Johan Vromans **<jvromans@squirrel.nl>**

1. Tcl Shell	5
2. Basic Tcl Language Features	5
3. Tcl Special Variables	6
4. Operators and Expressions	7
5. Regular Expressions	7
6. Pattern Globbing	8
7. Control Statements	9
8. File Information	9
9. Tcl Interpreter Information	11
10. Lists	12
11. Arrays	14
12. Strings and Binary Data	15
13. System Interaction	19
14. File Input/Output	20
15. Multiple Interpreters	23
16. Packages	24
17. Namespaces	25
18. Other Tcl Commands	26
19. Tk Shell	31
20. Tk Special Variables	31
21. General Tk Widget Information	31
22. Widget Scroll Commands	34
23. Entry Validation	35
24. The Canvas Widget	36
25. The Entry Widget	44
26. The Listbox Widget	46
27. The Menu Widget	47
28. The Text Widget	49
29. Other Standard Widgets	53
30. Images	63
31. Window Information	65
32. The Window Manager	67
33. Binding and Virtual Events	69
34. Geometry Management	71
35. Fonts	74
36. Other Tk Commands	74
37. TclX 8.4	79
38. TclX Special Variables and Commands	79
39. TclX General Commands	79
40. TclX Debugging Commands	81
41. TclX Development Commands	81
42. TclX Unix Access Commands	82
43. TclX File Commands	84
44. TclX Network Programming Support	86

45. TclX File Scanning Commands	86
46. TclX Math Commands	86
47. TclX List Manipulation Commands	87
48. TclX Keyed Lists	88
49. TclX String/Character Commands	88
50. TclX XPG/3 Message Catalog Commands	89
51. Img 1.2.4 Package	90
52. Tcllib 1.4	91
53. Tktable 2.8 Package	98
54. Vu 2.1.0 Package	105
55. Vim 6.2 if_tcl Interface	112

Conventions

fixed	denotes literal text.
<i>this</i>	means variable text, i.e. things you must fill in.
word	is a keyword, i.e. a word with a special meaning.
[...]	denotes an optional part.

1. Tcl Shell

tclsh [*script* [*arg* ...]]

Tclsh reads Tcl commands from its standard input or from file *script* and evaluates them.

Without arguments, **tclsh** runs interactively, sourcing the file **.tclshrc** (if it exists) before reading the standard input. With arguments, *script* is the name of a file to source and any additional arguments will become the value of the Tcl variable **\$argv**.

2. Basic Tcl Language Features

; or <i>newline</i>	statement separator
\	statement continuation if last character in line
#	comments out rest of line (if first non-whitespace character)
var	simple variable
var(index)	associative array variable
var(i,j)	multi-dimensional array variable
\$var	variable substitution (also \${var}xyz)
[expr 1+2]	command substitution
\char	backslash substitution (see below)
"hello \$a"	quoting with substitution
{hello \$a}	quoting with no substitution (deferred substitution)

The only data type in Tcl is a string. However, some commands will interpret arguments as numbers/boolean in which case the formats are

Integer: **123 0xff** (hex) **0377** (octal).

Floating Point: **2.1 3. 6e4 7.91e+16**

Boolean: **true false 0 1 yes no**

Tcl makes the following backslash substitutions:

\a audible alert (0x7)	\space space
\b backspace (0x8)	\newline space
\f form feed (0xC)	\ddd octal value (<i>d</i> =0–7)
\n newline (0xA)	\xdd hexadecimal value (<i>d</i> =0–9, a–f)
\r carriage return (0xD)	\udddd hexadecimal unicode value (<i>d</i> =0–9, a–f)
\t horizontal tab (0x9)	\c replace ‘\c’ with ‘c’
\v vertical tab (0xB)	\\ a backslash

3. Tcl Special Variables

argc	Number of command line arguments, not including the name of the script file.
argv	Tcl list (possibly empty) of command line arguments.
argv0	Name of script or command interpreter.
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	List of directories to search during auto-load operations. \$env(TCLLIBPATH) overrides the default list.
env	Array where each element name is an environment variable.
errorCode	Error code information from the last Tcl error. A list containing <i>class</i> [<i>code</i> [<i>msg</i>]].
errorInfo	Describes the stack trace of the last Tcl error.
tcl_library	Directory containing library of standard Tcl scripts. Value is returned by [info library]. \$env(TCL_LIBRARY) overrides the built-in value.
tcl_libPath	List of all possible locations for Tcl packages.
tcl_interactive	True if command interpreter is running interactively.
tcl_patchLevel	Integer specifying current patch level for Tcl.
tcl_pkgPath	List of directories to search for installed packages. Value is added to \$auto_path .
tcl_platform	Array with elements byteOrder , debug (if compiled with debugging enabled), machine , os , osVersion , platform , threaded (if compiled with threads enabled), user , and wordSize .
tcl_precision	Number of significant digits to retain when converting floating-point numbers to strings (default 12).
tcl_prompt1	A script which, when evaluated, outputs the main command prompt during interactive execution of tclsh or wish .
tcl_prompt2	A script which, when evaluated, outputs the secondary command prompt during interactive execution of tclsh or wish .
tcl_rcFileName	Name of script to source upon start-up of an interactive tclsh or wish .
tcl_traceCompile	Level of tracing info output during bytecode compilation: 0 for none, 1 for summary line, or 2 for byte code instructions.
tcl_traceExec	Level of tracing info output during bytecode execution: 0 for none, 1 for summary line, 2 and 3 for detailed trace.
tcl_wordchars	If set, a regular expression that controls what are considered to be “word” characters.
tcl_nonwordchars	If set, a regular expression that controls what are considered to be “non-word” characters.
tcl_version	Current version of Tcl in <i>major.minor</i> form.

4. Operators and Expressions

The **expr** command recognizes the following operators, in decreasing order of precedence:

-	~	!	unary minus, bitwise NOT, logical NOT	
*	/	%	multiply, divide, remainder	
+	-		add, subtract	
<<	>>		bitwise shift left, bitwise shift right	
<	>	<=	>=	boolean comparisons
==	!=			boolean equals, not equals
eq	ne			boolean string equals, not equals
&				bitwise AND
^				bitwise exclusive OR
 				bitwise inclusive OR
&&				logical AND
 				logical OR
x ? y : z				if x != 0 , then y , else z

All operators support integers. All support floating point except **~**, **%**, **<<**, **>>**, **&**, **^**, and **|**. Boolean operators can also be used for string operands (but note **eq** and **ne**), in which case string comparison will be used. This will occur if any of the operands are not valid numbers. The **&&**, **||**, and **? :** operators have “lazy evaluation”, as in C.

Possible operands are numeric values, Tcl variables (with **\$**), strings in double quotes or braces, Tcl commands in brackets, and the following math functions:

abs	cos	hypot	round	tanh
acos	cosh	int	sin	wide
asin	double	log	sinh	
atan	exp	log10	sqrt	
atan2	floor	pow	srand	
ceil	fmod	rand	tan	

5. Regular Expressions

regex*	match zero or more of <i>regex</i>
regex+	match one or more of <i>regex</i>
regex?	match zero or one of <i>regex</i>
regex{m}	match exactly <i>m</i> of <i>regex</i>
regex{m,}	match <i>m</i> or more of <i>regex</i>
regex{m,n}	match <i>m</i> through <i>n</i> of <i>regex</i>
*? +? ?? {m}? {m,}? {m,n}?	match smallest number of <i>regex</i> rather than longest
regex regex	match either expression
.	any single character except newline
()	capture group (possibly empty)
(?:)	non-capture group (possibly empty)
^ \$	match beginning, end of line
(?=regex) (?!regex)	match ahead to point where <i>regex</i> begins, does not begin
[abc] [^abc]	match characters in set, not in set
[a-z] [^a-z]	match characters in range, not in range

<code>[<i>class</i>:]</code>	within range, match <i>class</i> character; <i>class</i> is alpha , upper , lower , digit , xdigit , alnum , print , blank , space , punct , graph , or cntrl
<code>\d</code> <code>\s</code> <code>\w</code>	synonym for <code>[[:digit:]]</code> , <code>[[:space:]]</code> , <code>[[:alnum:]]_</code>
<code>\D</code> <code>\S</code> <code>\W</code>	synonym for <code>[^[:digit:]]</code> , <code>[^[:space:]]</code> , <code>[^[:alnum:]]_</code>
<code>\c</code>	match character <i>c</i> even if special
<code>\cX</code>	match control character [^] <i>X</i>
<code>\0</code> <code>\xhhh</code>	match character with value 0, value 0xhhh
<code>\e</code>	match ESC
<code>\B</code>	synonym for <code>\</code>
<code>\A</code> <code>\Z</code>	match beginning, end of string
<code>\m</code> <code>\M</code>	match beginning, end of word
<code>\y</code>	match beginning or end of word
<code>\Y</code>	match at point which is not beginning or end of word
<code>\m</code>	back reference to <i>m</i> th group

Meta syntax is specified when the *regex* begins with a special form.

<code>***=<i>regex</i></code>	<i>regex</i> is a literal string
<code>(?<i>letters</i>)<i>regex</i></code>	Embedded options: <ul style="list-style-type: none"> b <i>regex</i> is a basic regular expression c case-sensitive matching e <i>regex</i> is an extended regular expression i case-insensitive matching (-nocase) n new-line sensitive matching (-line) p partial new-line sensitive matching (-linestop) q rest of <i>regex</i> is a literal s non-newline sensitive matching (default) t tight syntax <i>regex</i> w inverse partial new-line sensitive matching (-lineanchor) x expanded syntax <i>regex</i> (-expanded)

6. Pattern Globbing

<code>?</code>	match any single character
<code>*</code>	match zero or more characters
<code>[abc]</code>	match set of characters
<code>[a-z]</code>	match range of characters
<code>\c</code>	match character <i>c</i>
<code>{a,b,...}</code>	match any of strings a, b, etc.
<code>~</code>	home directory (for glob command)
<code>~user</code>	match <i>user</i> 's home directory (for glob command)

Note: for the **glob** command, a `'` at the beginning of a file's name or just after `'` must be matched explicitly and all `'` characters must be matched explicitly.

7. Control Statements

break Abort innermost containing loop command.

case Obsolete, see **switch**.

continue

Skip to the next iteration of innermost containing loop command.

exit [*returnCode*]

Terminate the process, returning *returnCode* (an integer which defaults to 0) to the system as the exit status.

for *start test next body*

Looping command where *start*, *next*, and *body* are Tcl command strings and *test* is an expression string to be passed to **expr** command.

foreach *varname list body*

The Tcl command string *body* is evaluated for each item in the string *list* where the variable *varname* is set to the item's value.

foreach *varlist1 list1 [varlist2 list2 ...] body*

Same as above, except during each iteration of the loop, each variable in *varlistN* is set to the current value from *listN*.

if *expr1 [then] body1 [elseif expr2 [then] body2 ...] [[else] bodyN]*

If expression string *expr1* evaluates true, Tcl command string *body1* is evaluated. Otherwise if *expr2* is true, *body2* is evaluated, and so on. If none of the expressions evaluate to true then *bodyN* is evaluated.

return [**-code** *code*] [**-errorinfo** *info*] [**-errorcode** *value*] [*string*]

Return immediately from current procedure with *string* as return value. The *code* is one of **ok** (default), **error** (**-errorinfo** and **-errorcode** provide values for the corresponding Tcl variables), **return**, **break**, **continue**, or *integer*.

switch [*options*] *string pattern1 body1 [pattern2 body2 ...]*

switch [*options*] *string {pattern1 body1 [pattern2 body2 ...] }*

The *string* argument is matched against each of the *pattern* arguments in order. The *bodyN* of the first match found is evaluated. If no match is found and the last pattern is the keyword **default**, its *bodyN* is evaluated. Possible options are **-glob**, **-regexp**, and **-exact** (default).

while *test body*

Evaluates the Tcl command string *body* as long as expression string *test* evaluates to true.

8. File Information

file atime *fileName [time]*

Query or set time *fileName* was last accessed as seconds since Jan. 1, 1970.

file attributes *fileName [option [value ...]]*

Query or set platform-specific attributes of *fileName*. Options are for UNIX: **-group** *id* | *name*, **-owner** *id* | *name*, **-permissions** *octal* | [**ugo**]?[**+-=**][**rxwst**], [...]; for Windows **-archive**, **-hidden**, **-longname**, **-readonly**, **-shortname**, **-system**; and for MacOS: **-creator**, **-hidden**, **-readonly**, **-type**.

file channels [*pattern*]

Returns list of open channels matching glob *pattern* in current interpreter. If no pattern is given, returns all channels.

file copy [-force] [- -] *source* [*source* ...] *target*
 Makes a copy of *source* under name *target*. If multiple sources are given, *target* must be a directory. Use **-force** to overwrite existing files.

file delete [-force] [- -] *fileName* [*fileName* ...]
 Removes given files. Use **-force** to remove non-empty directories.

file dirname *fileName*
 Returns all directory path components of *fileName*.

file executable *fileName*
 Returns 1 if *fileName* is executable by user, 0 otherwise.

file exists *fileName*
 Returns 1 if *fileName* exists (and user can read its directory), 0 otherwise.

file extension *fileName*
 Returns all characters in *fileName* after and including the last dot.

file isdirectory *fileName*
 Returns 1 if *fileName* is a directory, 0 otherwise.

file isfile *fileName*
 Returns 1 if *fileName* is a regular file, 0 otherwise.

file join *name* [*name* ...]
 Joins file names using the correct path separator for the current platform.

file link *linkName*
 Return target of symbolic link *linkName*.

file link *linkName target*
 Create symbolic link *linkName* to *target*.

file link [-hard] *linkName* [*target*]
file link [-symbolic] *linkName* [*target*]
 Create either hard or symbolic link *linkName* to *target*.

file lstat *fileName* *varName*
 Same as **file stat** except uses the lstat kernel call.

file mkdir *dirName* [*dirName* ...]
 Creates given directories.

file mtime *fileName* [*time*]
 Query or set time *fileName* was last modified as seconds since Jan. 1, 1970.

file nativeName *fileName*
 Returns the platform-specific name of *fileName*.

file normalize *fileName*
 Returns normalised path representation for *fileName*.

file owned *fileName*
 Returns 1 if *fileName* owned by the current user, 0 otherwise.

file pathtype *fileName*
 Returns one of **absolute**, **relative**, or **volumerelative**.

file readable *fileName*
 Returns 1 if *fileName* is readable by current user, 0 otherwise.

file readlink *fileName*
 Returns value of symbolic link given by *fileName*.

file rename [-force] [- -] *source* [*source* ...] *target*
 Renames file *source* to *target*. If *target* is an existing directory, each source file is moved there. The **-force** option forces overwriting of existing files.

file rootname *fileName*
 Returns all the characters in *fileName* up to but not including last dot.

file separator

Returns path separator for the current platform.

file separator *[fileName]*

Returns path separator for filesystem containing *fileName*.

file size *fileName*

Returns size of *fileName* in bytes.

file split *fileName*

Returns list whose elements are the path components of *fileName*.

file stat *fileName* *varName*

Place results of stat kernel call on *fileName* in variable *varName* as an array with elements **atime**, **ctime**, **dev**, **gid**, **ino**, **mode**, **mtime**, **nlink**, **size**, **type**, and **uid**.

file system *fileName*

Returns two-element list for *fileName*: filesystem name and type (which may be null).

file tail *fileName*

Return all characters in *fileName* after last directory separator.

file type *fileName*

Returns type of *fileName*. Possible values are **file**, **directory**, **characterSpecial**, **blockSpecial**, **fifo**, **link**, or **socket**.

file volumes

Returns just ‘/’ on UNIX, list of local drives on Windows, and list of local and network drives on MacOS.

file writable *fileName*

Returns 1 if *fileName* is writable by current user, 0 otherwise.

9. Tcl Interpreter Information

info args *procName*

Returns list describing in order the names of arguments to *procName*.

info body *procName*

Returns the body of procedure *procName*.

info cmdcount

Returns the total number of commands that have been invoked.

info commands *[pattern]*

Returns list of Tcl commands (built-ins and procs) matching glob *pattern* (default *). If no pattern is given, returns all commands in current namespace.

info complete *command*

Returns 1 if *command* is a complete Tcl command, 0 otherwise. Complete means having no unclosed quotes, braces, brackets or array element names

info default *procName* *arg* *varName*

Returns 1 if procedure *procName* has a default for argument *arg* and places the value in variable *varName*. Returns 0 if there is no default.

info exists *varName*

Returns 1 if the variable *varName* exists in the current context, 0 otherwise.

info functions *[pattern]*

Returns list of math functions matching glob *pattern* (default *).

info globals *[pattern]*

Returns list of global variables matching glob *pattern* (default *).

info hostname

Returns name of computer on which interpreter was invoked.

info level

Returns the stack level of the invoking procedure.

info level *number*

Returns name and arguments of procedure invoked at stack level *number*.

info library

Returns name of library directory where standard Tcl scripts are stored.

info loaded [*interp*]

Returns list describing packages loaded into *interp*.

info locals [*pattern*]

Returns list of local variables matching glob *pattern* (default ***).

info nameofexecutable

Returns full pathname of binary from which the application was invoked.

info patchlevel

Returns current patch level for Tcl.

info procs [*pattern*]

Returns list of Tcl procedures in current namespace matching glob *pattern* (default ***).

info script [*filename*]

Query or set name of Tcl script currently being evaluated.

info sharedlibextension

Returns extension used by platform for shared objects.

info tclversion

Returns version number of Tcl in *major.minor* form.

info vars [*pattern*]

Returns list of currently-visible variables matching glob *pattern* (default ***).

10. Lists

concat [*arg arg ...*]

If all *args* are lists, return a list which is a concatenation of each *arg*.
Otherwise, return the concatenation of the string value of each *arg* (separated by a space) as a single string.

join *list* [*joinString*]

Returns string created by joining all elements of *list* with *joinString*.

lappend *varName* [*value value ...*]

Appends each *value* to the end of the list stored in *varName*.

lindex *list***lindex *list* {}**

Returns *list*.

lindex *list index*

Returns value of element at *index* in *list*.

lindex *list index ...***lindex *list indexList***

Each successive index value is used to select the element in the sublist selected by the previous index value. Return the result.

linsert *list index element* [*element ...*]

Returns new list formed by inserting given new elements at *index* in *list*.

list [*arg arg ...*]

Returns new list formed by using each *arg* as an element.

llength *list*

Returns number of elements in *list*.

lrange *list first last*

Returns new list from slice of *list* at indices *first* through *last* inclusive.

lreplace *list first last [value value ...]*

Returns new list formed by replacing elements *first* through *last* in *list* with given values.

lsearch [*options*] *list pattern*

Returns index of first element in *list* that matches *pattern* (-1 for no match).

Options are

-all	return all matching indices or values
-ascii	list elements are strings (only meaningful with -exact or -sorted)
-decreasing	list elements are sorted in decreasing order (only meaningful with -sorted)
-dictionary	list elements are compared using dictionary-style comparisons (only meaningful with -exact or -sorted)
-exact	string match
-glob	glob pattern match (default)
-increasing	list elements are sorted in increasing order (only meaningful with -sorted)
-inline	return matching value, not index
-integer	list elements are compared as integers (only meaningful with -exact or -sorted)
-not	negate the sense of the match
-real	list elements are compared as floating point values (only meaningful with -exact or -sorted)
-regexp	regex match
-sorted	use sorted list search algorithm (cannot be used with -glob or -regexp)
-start index	start search at <i>index</i>

lset *varName newValue*

lset *varName { } newValue*

Replace the value of list *varName* with *newValue*. Return the result.

lset *varName index newValue*

Replace the element at *index* in list *varName* with *newValue*. Return the result.

lset *varName index ... newValue*

lset *varName indexList newValue*

Replace the element identified by *index ...* (or *indexList*) in list *varName* with *newValue*. Each successive index value is used to select the element in the sublist selected by the previous index value. Return the result.

lsort [*switches*] *list*

Returns new list formed by sorting *list* according to *switches*. Switches are

-ascii	string comparison (default)
-dictionary	like -ascii but ignores case and is number smart.

-index <i>ndx</i>	treats each elements as a sub-list and sorts on the <i>ndx</i> th element
-integer	integer comparison
-real	floating-point comparison
-increasing	sort in increasing order (default)
-decreasing	sort in decreasing order
-command <i>cmd</i>	Use <i>cmd</i> which takes two arguments and returns an integer less than, equal to, or greater than zero
-unique	retain only the last duplicate element

split *string* [*splitChars*]

Returns a list formed by splitting *string* at each character in *splitChars* (default white-space).

List Indices:

Start at 0. The word **end** may be used to reference the last element in the list, and **end-integer** refers to the last element in the list minus the specified *integer* offset.

11. Arrays

array anymore *arrayName* *searchId*

Returns 1 if anymore elements are left to be processed in array search *searchId* on *arrayName*, 0 otherwise.

array donesearch *arrayName* *searchId*

Terminates the array search *searchId* on *arrayName*.

array exists *arrayName*

Returns 1 if *arrayName* is an array variable, 0 otherwise.

array get *arrayName*

Returns a list where each odd element is an element name and the following even element its corresponding value.

array names *arrayName* [*mode*] [*pattern*]

Returns list of all element names in *arrayName* that match *pattern*. Mode may be **-exact**, **-glob** (default *****), or **-regexp**.

array nextelement *arrayName* *searchId*

Returns name of next element in *arrayName* for the search *searchId*.

array set *arrayName* *list*

Sets values of elements in *arrayName* for list in **array get** format.

array size *arrayName*

Return number of elements in *arrayName*.

array startsearch *arrayName*

Returns a search id to use for an element-by-element search of *arrayName*.

array statistics *arrayName*

Returns hashtable statistics for *arrayName*.

array unset *arrayName* [*pattern*]

Unsets all of the elements in the array that match glob *pattern* (default *****). If *pattern* is omitted, the entire array is unset.

parray *arrayName* [*pattern*]

Print to standard output the names and values of all element names in *arrayName* that match glob *pattern* (default *****.)

12. Strings and Binary Data

append *varName* [*value value ...*]

Appends each of the given values to the string stored in *varName*.

binary format *formatString* [*arg arg ...*]

Returns a binary string representation of *argss* composed according to *formatString*, a sequence of zero or more field codes each followed by an optional integer count or *. The possible field codes are:

a	chars (null padding)	c	8-bit int	W	64-bit int (big)
A	chars (space padding)	s	16-bit int (little)	f	float
b	binary (low-to-high)	S	16-bit int (big)	d	double
B	binary (high-to-low)	i	32-bit int (little)	x	nulls
h	hex (low-to-high)	I	32-bit int (big)	X	backspace
H	hex (high-to-low)	w	64-bit int (little)	@	absolute position

binary scan *string formatString* [*varName varName ...*]

Extracts values into *varName*'s from binary *string* according to *formatString*. Returns the number of values extracted. For integer and floating point field codes, a list of *count* values are consumed and stored in the corresponding *varName*. Field codes are the same as for **binary format** with the additional:

a chars (no trimming) **A** chars (trimming) **x** skip forward

format *formatString* [*arg arg ...*]

Returns a formatted string generated in the ANSI C **sprintf**-like manner.

Place holders have the form `%[argpos$][flag][width][.prec][h|l]char` where *argpos* is an integer, *width* and *prec* are integers or *, possible values for *char* are:

d	signed decimal	x	unsigned HEX	E	float (0E0)
u	unsigned decimal	c	int to char	g	auto float (f or e)
i	signed decimal	s	string	G	auto float (f or E)
o	unsigned octal	f	float (fixed)	%	plain %
x	unsigned hex	e	float (0e0)		

and possible values for *flag* are:

-	left-justified	0	zero padding	#	alternate output
+	always signed	space	space padding		

The optional trailing **h** or **l** truncates the data to 16-bits or expands it to 64-bits for integer conversions.

regexp [*switches*] *exp string* [*matchVar*] [*subMatchVar ...*]

Returns 1 if the regular expression *exp* matches part or all of *string*, 0 otherwise. If specified, *matchVar* will be set to all the characters in the match and the following *subMatchVar*'s will be set to matched parenthesized subexpressions. Switches are

- about** Instead of matching, returns an informational list. The first element is a subexpression count, the second a list of property names that describe various attributes of the regular expression.
- all** Match as many times as possible in the string, returning the total number of matches found. If match

variables are present, they will continue information for the last match only.

-expanded	Enable expanded form of the regular expression.
-indices	<i>matchVar</i> and <i>subMatchVar</i> will be set to the start and ending indices in <i>string</i> of their corresponding match.
-inline	Return, as a list, the data that would otherwise be placed in match variables (which may not be specified.) If used with -all , the list will be concatenated at each iteration. For each match iteration, the command will append the overall match data, plus one element for each subexpression in the regular expression.
-line	Enable newline-sensitive matching. With this flag, '[' bracket expressions and '.' never match newline, '^' matches an empty string after any newline in addition to its normal function, and '\$' matches an empty string before any newline in addition to its normal function. This flag is equivalent to specifying both -linestop and -lineanchor .
-lineanchor	With this flag, '^' and '\$' match the beginning and end of a line respectively.
-linestop	With this flag, '[' bracket expressions and '.' stop at newlines.
-nocase	Ignore case in matching.
-start index	An offset into the string to start matching.

regsub [*switches*] *exp string subSpec* [*varName*]

Replaces the first portion of *string* that matches the regular expression *exp* with *subSpec* and optionally places results in *varName*. If *subSpec* contains a '&' or "\0", then it is replaced in the substitution with the portion of string that matched *exp*. If *subSpec* contains a "\n" then it is replaced in the substitution with the portion of string that matched the *n*th parenthesized subexpression of *exp*. Returns either count of number of replacements made if *varName* was provided or the result of the substitution. Switches are

-all	Substitution is performed for each ranges in <i>string</i> that match.
-expanded	Enable expanded form of the regular expression.
-indices	<i>matchVar</i> and <i>subMatchVar</i> will be set to the start and ending indices in <i>string</i> of their corresponding match.
-line	Enable newline-sensitive matching. With this flag, '[' bracket expressions and '.' never match newline, '^' matches an empty string after any newline in addition to its normal function, and '\$' matches an empty string before any newline in addition to its normal function. This flag is equivalent to specifying both -linestop and -lineanchor .
-lineanchor	With this flag, '^' and '\$' match the beginning and end of a line respectively.
-linestop	With this flag, '[' bracket expressions and '.' stop at newlines.
-nocase	Ignore case in matching.

-start *index* An offset into the string to start matching.

scan *string formatString* [*varName varName ...*]

Extracts values into given variables using ANSI C **sscanf** behavior. Returns the number of values extracted, or if no *varName*'s were specified, returns a list of the data that would otherwise be stored in the variables. Place holders have the form **%[*][width][h|l|L]char** where ***** is for discard, *width* is an integer. If the form is **%integer\$char**, then the variable to use is taken from the argument indicated by the number, where 1 corresponds to the first *varName*. If there are any positional specifiers in format then all of the specifiers must be positional. Possible values for *char* are:

d	decimal	e	float	s	string (non-whitespace)
o	octal	f	float	[chars]	chars in given range
x	hex	g	float	[^chars]	chars not in given range
u	unsigned	i	integer	c	char to int
n	number scanned				

The modifier **h** is ignored, but **l** or **L** scans a 64-bit value for integer conversions.

string bytelength *string*

Returns the number of bytes to represent the UTF-8 representation of *string* in memory.

string compare [*options*] *string1 string2*

Returns -1, 0, or 1, depending on whether *string1* is lexicographically less than, equal to, or greater than *string2*. Options are **-nocase** or **-length integer**.

string equal [*options*] *string1 string2*

Returns 0 or 1, depending on whether *string1* is lexicographically equal to *string2*. Options are **-nocase** or **-length integer**.

string first *string1 string2* [*startIndex*]

Return index (-1 if not found) in *string2* of first occurrence of *string1* starting at *startIndex* (default 0).

string index *string charIndex*

Returns for *index* an *integer*, the *charIndex*'th character in *string*. If *index* is **end**, the last character in *string*, and if *index* is **end-integer**, the last character minus the specified offset.

string is *class* [*options*] *string*

Returns 1 if *string* is a valid member of the specified character class, 0 otherwise. If option **-strict** is specified, then an empty string returns 0, otherwise 1 on any class. If **-failindex** *varname* is specified then, on failure, the index in the *string* where the *class* was no longer valid will be stored in *varname*. Character classes are

alnum	Alphabetic or digit character
alpha	Alphabetic character
ascii	Character in 7-bit ascii range
boolean	Any of Tcl boolean forms
control	A control character
digit	A digit character
double	Any of Tcl double forms with optional surrounding white space; 0 returned for under/overflow with <i>varname</i> set to -1
false	Any Tcl boolean false form

graph	Any printing character except space
integer	Any of Tcl integer forms with optional surrounding white space; 0 returned for under/overflow with <i>varname</i> set to -1
lower	A lower case alphabetic character
print	Any printing character including space
punct	A punctuation character
space	A white-space character
true	Any Tcl boolean true form
upper	A upper case character
wordchar	A valid word character
xdigit	A digit or letter in the range [a-fA-F]

string last *string1 string2 lastIndex*

Return index (-1 if not found) in *string2* of last occurrence of *string1* no higher than *lastIndex* (default **end**).

string length *string*

Returns the number of characters in *string*.

string map [**-nocase**] *charMap string*

Replaces characters in *string* based on the key-value pairs in *charMap*. Each instance of a key in the string will be replaced with its corresponding value.

string match [**-nocase**] *pattern string*

Returns 1 if glob *pattern* matches *string*, 0 otherwise.

string range *string first last*

Returns characters from *string* at indices *first* through *last* inclusive. The indices are as for **string index**.

string repeat *string count*

Returns new string formed by *string* repeated *count* times.

string replace *string first last [newstring]*

Removes characters from *string* at indices *first* through *last* inclusive. If *newstring* is supplied, it will replace the removed characters. The indices are as for **string index**.

string tolower *string [first [last]]*

Returns new string formed by converting all chars in *string* to lower case at indices *first* through *last* inclusive. The indices are as for **string index**.

string totitle *string [first [last]]*

Returns new string formed by converting the first character in *string* to upper case and the remainder to lower case at indices *first* through *last* inclusive. The indices are as for **string index**.

string toupper *string [first [last]]*

Returns new string formed by converting all chars in *string* to upper case at indices *first* through *last* inclusive. The indices are as for **string index**.

string trim *string [chars]*

Returns new string formed by removing from *string* any leading or trailing characters present in the set *chars* (default whitespace).

string trimleft *string [chars]*

Same as **string trim** for leading characters only.

string trimright *string [chars]*

Same as **string trim** for trailing characters only.

string wordend *string index*

Returns index of character just after last one in word at *index* in *string*.

string wordstart *string index*

Returns index of first character of word at *index* in *string*.

subst [-noblackslashes] [-nocommands] [-novariables] *string*

Returns result of backslash, command, and variable substitutions on *string*.

Each may be turned off by switch.

13. System Interaction

cd [*dirName*]

Change working directory to *dirName*.

clock clicks [-**milliseconds**]

Returns hi-res system-dependent integer time value. If **-milliseconds** is specified, then the value is guaranteed to be of millisecond granularity.

clock format *clockVal* [-**format** *string*] [-**gmt** *boolean*]

Convert integer *clockVal* to human-readable format defined by *string* which recognizes (at least) the following place holders:

%%	%	%p	AM/PM
%a	weekday (abbr)	%Q	Stardate
%A	weekday (full)	%r	locale time; %I:%M:%S %p
%b	month (abbr)	%R	%H:%M
%B	month (full)	%S	seconds (00 – 59)
%C	century (19 or 20)	%s	seconds since epoch
%d	day (01 – 31)	%t	\t
%D	%m/%d/%y	%T	%H:%M:%S
%e	day (1 – 31)	%U	week, Sun–Sat (00 – 52)
%H	hour (00 – 23)	%u	weekday (Mon=1, Sun=7)
%h	month (abbr)	%V	week, Jan 4 is week 1 (01 – 53)
%I	hour (01 – 12)	%W	week, Mon–Sun (00 – 52)
%j	day (001 – 366)	%w	weekday (Sun=0 – Sat=6)
%k	hour (0 – 23)	%x	locale date; %m/%d/%y
%l	hour (1 – 12)	%X	locale time; %H:%M:%S
%M	minute (00 – 59)	%y	year (00 – 99)
%m	month (01 – 12)	%Y	year (full)
%n	\n	%Z	time zone
%c	locale date & time; %a %b %d %H:%M:%S %Y		

The default format is "%a %b %d %H:%M:%S %Z %Y".

clock scan *dateString* [-**base** *clockVal*] [-**gmt** *boolean*]

Convert *dateString* to an integer clock value. If *dateString* contains a 24 hour time only, the date given by *clockVal* is used.

clock seconds

Return current date and time as system-dependent integer value.

exec [-**keepnewline**] *arg* [*arg* ...]

Execute subprocess using each *arg* as word for a shell pipeline and return results written to standard out, optionally retaining the final newline char.

The following constructs can be used to control I/O flow.

	pipe (stdout)
&	pipe (stdout and stderr)
< <i>fileName</i>	stdin from file

<@ <i>fileId</i>	stdin from open file
<< <i>value</i>	pass value to stdin
> <i>fileName</i>	stdout to file
2> <i>fileName</i>	stderr to file
>& <i>fileName</i>	stdout and stderr to file
>> <i>fileName</i>	append stdout to file
2>> <i>fileName</i>	append stderr to file
>>& <i>fileName</i>	stdout and stderr to file
>@ <i>fileId</i>	stdout to open file
2>@ <i>fileId</i>	stderr to open file
>&@ <i>fileId</i>	stdout and stderr to open file
&	run in background

glob [*switches*] *pattern* [*pattern* ...]

Returns list of all files in current directory that match any of the given glob patterns, as interpreted by the given switches. Switches are

-directory <i>directory</i>	Search starting in the given <i>directory</i> . May not be used with -path .
-join	The pattern arguments are treated as a single <i>pattern</i> obtained by joining the arguments with directory separators.
-nocomplain	Allows an empty list to be returned without error.
-path <i>pathPrefix</i>	Search with the given <i>pathPrefix</i> where the rest of the name matches the given <i>patterns</i> . May not be used with -directory .
-tails	Only return part of file which follows last directory named in -directory or -path specification.
-types <i>typeList</i>	Only match files or directories which match <i>typeList</i> , where the items in the list have two forms (which may be mixed.) The first form is: b (block special file), c (character special file), d (directory), f (plain file), l (symbolic link), p (named pipe), or s (socket), where multiple types may be specified in the list. Return all files which match at least one of the types given. For the second form, all the types must match. These are r , w , x as file permissions, and readonly , hidden as special permission cases.

pid [*fileId*]

Return process id of process pipeline *fileId* if given, otherwise return process id of interpreter process.

pwd Returns the current working directory.

14. File Input/Output

close *fileId*

Close the open file channel *fileId*.

eof *fileId*

Returns 1 if an end-of-file has occurred on *fileId*, 0 otherwise.

fblocked *fileId*

Returns 1 if last read from *fileId* exhausted all available input.

fconfigure *fileId* [*option* [*value*]]

Sets and gets options for I/O channel *fileId*. Options are:

-blocking *boolean* Whether I/O can block process.

-buffering **full** | **line** | **none** How to buffer output.

-buffersize *byteSize* Size of buffer.

-encoding *name*
Specify Unicode encoding to *name* or **binary**.

-eofchar *char* | {*inChar* *outChar*}
Sets character to serve as end-of-file marker.

-translation *mode* | {*inMode* *outMode*}
Sets how to translate end-of-line markers.
Modes are **auto**, **binary**, **cr**, **crlf**, and **lf**.

For socket channels (read-only settings):

-error
Returns current error status.

-sockname
Returns three element list with address, host name and port number.

-peername
For client and accepted sockets, three element list of peer socket.

For serial device channels:

-mode *baud,parity,data,stop*
Set baud rate, parity (**n**, **o**, **e**, **m** or **s**), data bits, and stop bits of channel.

-handshake *type*
Set handshake control (**none**, **rtscts** or **xonxoff**).

-queue
Return a list of two integers: current number of bytes in input queue and output queue.

-timeout *msec*
Set timeout for blocking reads.

-ttycontrol {*signal* *boolean* ...}
Setup the handshake output lines (**rts** or **dtr**) permanently or send a BREAK (**break**).

-ttystatus
Return list of current modem status and handshake signals: {**CTS** *boolean* **DSR** *boolean* **RING** *boolean* **DCD** *boolean* }

-xchar {*xonChar* *xoffChar*}
Set software handshake characters.

fcopy *inId* *outId* [**-size** *size*] [**-command** *callback*]

Transfer data to *outId* from *inId* until eof or *size* bytes have been transferred. If **-command** is given, copy occurs in background and runs *callback* when finished appending number of bytes copied and possible error message as arguments.

fileevent *fileId* **readable** | **writable** [*script*]

Evaluate *script* when channel *fileId* becomes readable/writable.

flush *fileId*

Flushes any output that has been buffered for *fileId*.

gets *fileId*

Read and return next line from channel *fileId*, discarding newline character.

gets *fileId* *varName*

Read next line from channel *fileId*, discarding newline character, into *varName*. Return count of characters read, or -1 if end-of-file.

open *item* [*access* [*perms*]]

Open *item* (a file, serial port or command pipeline) and return its channel id.

A command pipeline (the first character of *item* is '|') is a list as described for **exec**. If a new file is created, its permission are set to the conjunction of *perms* (default 0666) and the process umask. The first form of *access* may be

r Read only. File must exist.

r+

Read and write. File must exist.

w Write only. Truncate if exists.

w+

Read and write. Truncate if exists.

a Write only. File must exist. Access position at end.

a+

Read and write. Access position at end.

The second form of *access* may be a list containing any of the following flags (although one of the flags must be either **RONLY**, **WRONLY** or **RDWR**.)

RONLY Open the file for reading only.

WRONLY Open the file for writing only.

RDWR Open the file for both reading and writing.

APPEND Set the file pointer to the end of the file prior to each write.

CREAT Create the file if it doesn't already exist.

EXCL If **CREAT** is also specified, an error is returned if the file already exists.

NOCTTY Prevent the terminal device from becoming the controlling terminal of the process.

NONBLOCK Prevents the process from blocking while opening the file, and possibly in subsequent I/O operations.

TRUNC If the file exists it is truncated to zero length.

puts [**-nonewline**] [*fileId*] *string*

Write *string* to *fileId* (default **stdout**) optionally omitting newline char.

read [**-nonewline**] *fileId*

Read all remaining bytes from *fileId*, optionally discarding last character if it is a newline.

read *fileId* *numBytes*

Read *numBytes* bytes from *fileId*.

seek *fileId* *offset* [*origin*]

Change current access position on *fileId* to *offset* bytes from *origin* which may be **start** (default), **current**, or **end**.

socket [*option* ...] *host* *port*

Open a client-side TCP socket to server *host* on *port* (integer or service name). Options are:

-myaddr *addr*

Set network address of client (if multiple available).

-myport *port* Set connection port of client (if different from server).

-async Make connection asynchronous.

socket -server *command* [**-myaddr** *addr*] *port*

Open server TCP socket on *port* (integer or service name) invoking *command* once connected with three arguments: the channel, the address, and the port number. If *port* is zero, the operating system will choose a port number.

tell *fileId*

Return current access position in *fileId*.

15. Multiple Interpreters

interp alias *srcPath srcCmd*

Returns list whose elements are the *targetCmd* and *args* associated with the alias *srcCmd* in interpreter *srcPath*.

interp alias *srcPath srcCmd* { }

Deletes the alias *srcCmd* in interpreter *srcPath*.

interp alias *srcPath srcCmd targetPath targetCmd* [*arg* ...]

Creates an alias *srcCmd* in interpreter *srcPath* which when invoked will run *targetCmd* and *args* in the interpreter *targetPath*.

interp aliases [*path*]

Returns list of all aliases defined in interpreter *path*.

interp create [**-safe**] [- -] [*path*]

Creates a slave interpreter (optionally safe) named *path*.

interp delete *path* [*path* ...]

Deletes the interpreter(s) *path* and all its slave interpreters.

interp eval *path arg* [*arg* ...]

Evaluates concatenation of *args* as command in interpreter *path*.

interp exists *path*

Returns 1 if interpreter *path* exists, 0 otherwise.

interp expose *path hiddenCmd* [*exposedCmd*]

Make *hiddenCmd* in interpreter *path* exposed (optionally as *exposedCmd*).

interp hide *path exposedCmd* [*hiddenCmd*]

Make *exposedCmd* in interpreter *path* hidden (optionally as *hiddenCmd*).

interp hidden *path*

Returns list of hidden commands in interpreter *path*.

interp invokehidden *path* [**-global**] *hiddenCmd* [*arg* ...]

Invokes *hiddenCmd* with specified *args* in interpreter *path* (at the global level if **-global** is given).

interp issafe [*path*]

Returns 1 if interpreter *path* is safe, 0 otherwise.

interp marktrusted [*path*]

Marks interpreter *path* as trusted.

interp recursionlimit *path* [*newLimit*]

Queries or sets recursion limit (default 1000) for interpreter *path*.

interp share *srcPath fileId destPath*

Arranges for I/O channel *fileId* in interpreter *srcPath* to be shared with interpreter *destPath*.

interp slaves [*path*]

Returns list of names of all slave interpreters of interpreter *path*.

interp target *path alias*

Returns Tcl list describing target interpreter of *alias* in interpreter *path*.

interp transfer *srcPath fileId destPath*

Moves I/O channel *fileId* from interpreter *srcPath* to *destPath*.

For each slave interpreter created, a new Tcl command is created by the same name in its master. This command has the **alias**, **aliases**, **eval**, **expose**, **hide**, **hidden**, **invokehidden**, **issafe**, **marktrusted** and **recursionlimit** subcommands like **interp**, but without the *srcPath* and *path* arguments (they default to the slave itself) and without the *targetPath* argument (it defaults to the slave's master).

A safe interpreter is created with the following commands exposed:

after	eval	info	package	string
append	expr	interp	pid	subst
array	fblocked	join	proc	switch
binary	fcopy	lappend	puts	tell
break	fileevent	index	read	time
case	flush	linsert	regexp	trace
catch	for	list	regsub	unset
clock	foreach	llength	rename	update
close	format	lrange	return	uplevel
concat	gets	lreplace	scan	upvar
continue	global	lsearch	seek	variable
eof	if	lsort	set	vwait
error	incr	namespace	split	while

A safe interpreter is created with the following commands hidden:

cd	exit	glob	open	socket
encoding	fconfigure	load	pwd	source
exec	file			

The following support procedures are also hidden:

auto_exec_ok	auto_import	auto_load
auto_load_index	auto_qualify	unknown

The **\$env** variable is also not present in a safe interpreter.

16. Packages

package forget *package* ...

Remove all info about each *package* from interpreter.

package ifneeded *package version* [*script*]

Tells interpreter that if version *version* of *package*, evaluating *script* will provide it.

package names

Returns list of all packages in the interpreter that are currently provided or have an **ifneeded** script available.

package provide *package* [*version*]

Tells interpreter that *package version* is now provided. Without *version*, the currently provided version of *package* is returned.

package require [**-exact**] *package* [*version*]

Tells interpreter that *package* must be provided. Only packages with versions equal to or later than *version* (if provided) are acceptable. If **-exact** is specified, the exact version specified must be provided.

package unknown [*command*]

Specifies a last resort Tcl command to provide a package which have append as its final two arguments the desired package and version.

package vcompare *version1 version2*

Returns -1 if *version1* is earlier than *version2*, 0 if equal, and 1 if later.

package versions *package*

Returns list of all versions numbers of *package* with an **ifneeded** script.

package vsatisfies *version1 version2*

Returns 1 if *version2* scripts will work unchanged under *version1*, 0 otherwise.

pkg_mkIndex [*switches*] *directory* [*pattern* ...]

Build the **pkgIndex.tcl** file for automatic loading of packages. Each glob *pattern* (default ***.tcl *.*[info sharedlibextension]***) selects script or binary files in *directory*. Switches are:

-direct (Default) The generated index will implement direct loading of the package upon **package require**.

-lazy The generated index will delay loading until the use of one of the commands provided by the package,

-load *pkgPat*

The index process will pre-load any packages that exist in the current interpreter and match glob *pkgPat* into the slave interpreter used to generate the index.

-verbose Generate output during the indexing process.

::pkg::create -name *pkgName* **-version** *version* *options* ...

Create appropriate **package ifneeded** command for version *version* of package *pkgName*. Options are:

-load *filespecList*

A binary library that must be loaded. The *filespecList* contains one or two elements: the first is the name of the file to load, the second (optional) is a list of commands supplied by loading that file.

-source *filespecList*

A Tcl library that must be sourced. The *filespecList* contains one or two elements: the first is the name of the file to load, the second (optional) is a list of commands supplied by loading that file.

17. Namespaces

namespace children [*namespace*] [*pattern*]

Returns list of child namespaces belonging to *namespace* (defaults to current) which match *pattern* (default *).

namespace code *script*

Returns new script string which when evaluated arranges for *script* to be evaluated in current namespace. Useful for callbacks.

namespace current

Returns fully-qualified name of current namespace.

namespace delete [*namespace* ...]

Each given namespace is deleted along with their child namespaces, procedures, and variables.

namespace eval *namespace arg* [*arg* ...]

Activates *namespace* and evaluates concatenation of *args*'s inside it.

namespace exists *namespace*

Returns 1 if *namespace* is a valid namespace in the current context, returns 0 otherwise.

namespace export [-clear] [*pattern* ...]

Adds to export list of current namespace all commands that match given *pattern*'s. If **-clear** is given, the export list is first emptied.

namespace forget [*namespace::pattern* ...]

Removes from current namespace any previously imported commands from *namespace* that match *pattern*.

namespace import [-force] [*namespace::pattern* ...]

Imports into current namespace commands matching *pattern* from *namespace*. The **-force** option allows replacing of existing commands.

namespace inscope *namespace listArg* [*arg* ...]

Activates *namespace* (which must already exist) and evaluates inside it the result of lappend of *arg*'s to *listArg*.

namespace origin *command*

Returns fully-qualified name of imported *command*.

namespace parent [*namespace*]

Returns fully-qualified name of parent namespace of *namespace*.

namespace qualifiers *string*

Returns any leading namespace qualifiers in *string*.

namespace tail *string*

Returns the simple name (strips namespace qualifiers) in *string*.

namespace which [-command | -variable] *name*

Returns fully-qualified name of the command (or as variable, if **-variable** given) *name* in the current namespace. Will look in global namespace if not in current namespace.

variable [*name value* ...] *name* [*value*]

Creates one or more variables in current namespace (if *name* is unqualified) initialized to optionally given *values*. Inside a procedure and outside a **namespace eval**, a local variable is created linked to the given namespace variable.

18. Other Tcl Commands

after *ms* [*arg1 arg2 arg3* ...]

Arrange for command (concat of *args*) to be run after *ms* milliseconds have passed. With no *args*, program will sleep for *ms* milliseconds. Returns the id of the event handler created.

after cancel *id* [*arg1 arg2* ...]

Cancel previous **after** command either by command or the id returned.

after idle [*arg1 arg2 arg3* ...]

Arrange for command (concat of *args*) to be run later when Tk is idle. Returns the id of the event handler created.

after info [*id*]

Returns information on event handler *id*. With no *id*, returns a list of all existing event handler ids.

auto_execok *execFile*

Returns full pathname if an executable file by the name *execFile* exists in user's PATH, empty string otherwise.

auto_load *command*

Attempts to load definition for *cmd* by searching **\$auto_path** and **\$env(TCLLIBPATH)** for a tclIndex file which will inform the interpreter where it can find *command*'s definition.

auto_mkindex *directory pattern [pattern ...]*

Generate a tclIndex file from all files in *directory* that match glob patterns.

auto_reset

Destroys cached information used by **auto_execok** and **auto_load**.

bgeerror *message*

User defined handler for background Tcl errors. Default exists for Tk which posts a dialog box containing the error message with an application configurable button (default is to save the stack trace to a file.) This behavior can be redefined with the global options:

*ErrorDialog.function.text	<i>buttonText</i>
*ErrorDialog.function.command	<i>tclProc</i>

If selected, *tclProc* is called with one argument: the text of the stack trace. If either of these options is set to the empty string, then the additional button will not be displayed in the dialog.

catch *script [varName]*

Evaluate *script* and store results into *varName*. If there is an error, a non-zero error code is returned and an error message stored in *varName*.

encoding convertfrom [*encoding*] *data*

Convert *data* to Unicode from the specified *encoding*. If *encoding* is not specified, the current system encoding is used.

encoding convertto [*encoding*] *string*

Convert *string* from Unicode to the specified *encoding*. If *encoding* is not specified, the current system encoding is used.

encoding names

Returns a list containing the names of all of the encodings that are currently available.

encoding system [*encoding*]

Query or set the system encoding.

error *message [info] [code]*

Interrupt command interpretation with an error described in *message*. Global variables **errorInfo** and **errorCode** will be set to *info* and *code*.

eval *arg [arg ...]*

Returns result of evaluating the concatenation of *args*'s as a Tcl command.

expr *arg [arg ...]*

Returns result of evaluating the concatenation of *arg*'s as an operator expression. See *Operators* for more info.

global *varName [varName ...]*

Declares given *varName*'s as global variables.

history add *command [exec]*

Adds *command* to history list, optionally executing it.

history change *newValue [event]*

Replaces value of *event* (default current) in history with *newValue*.

history clear

Erase the history list and reset event numbers.

history event [*event*]

Returns value of *event* (default -1) in history.

history info [*count*]

Returns event number and contents of the last *count* events.

history keep [*count*]

Set number of events to retain in history to *count*.

history nextid

Returns number for next event to be recorded in history.

history redo [*event*]

Re-evaluates *event* (default -1).

incr *varName* [*increment*]

Increment the integer value stored in *varName* by *increment* (default 1).

load *file* [*pkgName* [*interp*]]

Load binary code for *pkgName* (default derived from *file* name) from *file* (dynamic lib) into *interp*. The initialization procedure for *pkgName* is then called to incorporate it into *interp*.

load {} *pkgName* [*interp*]

Search for statically linked or previously loaded *pkgName*. The initialization procedure for *pkgName* is then called to incorporate it into *interp*.

proc *name* *args* *body*

Create a new Tcl procedure (or replace existing one) called *name* where *args* is a list of arguments and *body* Tcl commands to evaluate when invoked.

rename *oldName* *newName*

Rename command *oldName* so it is now called *newName*. If *newName* is the empty string, command *oldName* is deleted.

set *varName* [*value*]

Store *value* in *varName* if given. Returns the current value of *varName*.

source *fileName*

Read file *fileName* (up to '^Z' character) and evaluate its contents as a Tcl script.

tcl_endOfWord *str* *start*

Return index of the first end-of-word location after *start* in *str*.

tcl_findLibrary *basename* *version* *patch* *initScript* *envVarName* *varName*

Search for directory containing extension script library (one script of which is *initScript*) setting global *varName* with the result and sourcing *initScript*. If *varName* is not preset, search for *initScript* in either directory **\$env(*envVarName*)** or (relative to directory containing **info nameofexecutable**) *../lib/basenameversion*, *../lib/basenameversion*, *../library*, *../library*, *../basename(version|patch)/library* or *../basename(version|patch)/library*.

tcl_startOfNextWord *str* *start*

Return index of the first start-of-word location after *start* in *str*.

tcl_startOfPreviousWord *str* *start*

Return index of the first start-of-word location before *start* in *str*.

tcl_wordBreakAfter *str* *start*

Return index of the first word boundary after *start* in *str*.

tcl_wordBreakBefore *str* *start*

Return index of the first word boundary before *start* in *str*.

time *script* [*count*]

Call interpreter *count* (default 1) times to evaluate *script*. Returns string of the form "503 microseconds per iteration".

trace add command *procName opsList tclProc*

Arrange for *tclProc* to be executed whenever command *procName* is modified as specified by *opsList*, one or more of **rename** or **delete**. When triggered, *tclProc* is called with three arguments:

tclProc oldName newName op

newName is empty for a **delete** operation.

trace add execution *procName opsList tclProc*

Arrange for *tclProc* to be executed whenever command *procName* is modified as specified by *opsList*, one or more of **enter**, **leave**, **enterstep** or **leavestep**. When triggered, *tclProc* is called with arguments:

tclProc command [code result] op

command is the complete command being executed. For **enter** and **enterstep**, execution can be prevented by deleting *command*. For **leave** and **leavestep**, *code* is the result code and *result* is the result string.

trace add variable *varName opsList tclProc*

Arrange for *tclProc* to be executed whenever *varName* is accessed as specified by *opsList*, one or more of **array**, **unset**, **read** or **write**. When triggered, *tclProc* is called with three arguments:

tclProc name1 name2 op

Name1 and *name2* give the name(s) for the variable being accessed: if a scalar or if an entire array is being deleted and the trace was registered on the overall array then *name1* is the variable's name and *name2* is empty; if the variable is an array element then *name1* is the array name and *name2* is the index into the array. *Op* indicates what operation is being performed on the variable as defined above.

trace info *type name*

Return list for each trace *type* (**command**, **execution** or **variable**) set on *procName*. Each element of the list is the *opsList* and *tclProc* associated with the trace.

trace remove *type procName opsList tclProc*

Remove the trace *type* (**command**, **execution** or **variable**) set on *procName* for the operations *opsList* with *tclProc*.

trace variable *varName ops tclProc*

Obsolete. Same as **trace add variable** *varName ops tclProc*.

trace vdelete *varName ops tclProc*

Obsolete. Same as **trace remove variable** *varName ops tclProc*.

trace vinfo *varName*

Obsolete. Same as **trace info variable** *varName*.

unknown *cmdName [arg arg ...]*

Called when the Tcl interpreter encounters an undefined command name.

unset [**-nocomplain**] [- -] *varName [varName ...]*

Removes the given variables and arrays from scope. If **-nocomplain** is specified, any possible errors are suppressed.

update [**idletasks**]

Handle pending events. If **idletasks** is specified, only those operations normally deferred until the idle state are processed.

uplevel [*level*] *arg [arg ...]*

Evaluates concatenation of *arg*'s in the variable context indicated by *level*,

an integer that gives the distance up the calling stack. If *level* is preceded by '#', then it gives the distance down the calling stack from the global level.

upvar [*level*] *otherVar myVar [otherVar myVar ...]*

Makes *myVar* in local scope equivalent to *otherVar* at context *level* (see **uplevel**) so they share the same storage space.

vwait *varName*

Enter Tcl event loop until global variable *varName* is modified.

19. Tk Shell

wish [*arg* ...]

wish *fileName* [*arg* ...]

Wish reads Tcl commands from its standard input or from file *fileName* and evaluates them.

With no arguments or with a first argument that starts with '-', **wish** runs interactively, sourcing the file **.wishrc** (if it exists) before reading the standard input. Otherwise, an initial argument *fileName* is the name of a file to source.

The *name* of the application, which is used for purposes such as **send** commands, is taken from the **-name** option, from *fileName*, or from the command name by which **wish** was invoked.

The *class* of the application, which is used for purposes such as specifying options with a *resource manager* property or **.Xdefaults** file, is the same as its *name* except that the first letter is capitalized.

The following command line options are recognized:

-colormap new

Use a new private colormap instead of the default screen colormap.

-display display

X11 display for main window.

-geometry geometry

Initial geometry of main window.

-name name Title to be displayed for the main window, and the name of the interpreter for **send** commands.

-sync Execute all X server commands synchronously.

-use id Embed the main window for the application in the window whose identifier is *id*.

-visual visual

Visual to use for the window.

--

All remaining arguments become the value of **\$argv**.

20. Tk Special Variables

geometry	The value of the -geometry command line option.
tk_library	Directory containing library of standard Tk scripts.
tk_patchLevel	Integer specifying current patch level for Tk.
tk::Priv	Array containing information private to standard Tk scripts.
tk_strictMotif	When non-zero, Tk tries to adhere to Motif look-and-feel as closely as possible.
tk_textRedraw tk_textRelayout	Set by text widgets when they have debugging turned on.
tk_version	Current version of Tk in <i>major.minor</i> form.

21. General Tk Widget Information

All widget are created with

widget pathname [*option1 value1* [*option2* ...]]

where *widget* is the Tcl command corresponding to the class of widget desired (eg. **button**) and *pathname* is a string which will be used to identify the newly created widget. In general, a widget name is the concatenation of its parent's name followed by a period (unless the parent is the root window '.')

and a string containing no periods (eg. **.mainframe.btnframe.btn1**).

Widget configuration options may be passed in the creation command. Options begin with a '-' and are always followed by a value string. After creation, options may be changed using the **configure** widget command

pathname **configure** *option1 value1* [*option2 ...*]

and queried using the **cget** command

pathname **cget** *option*

Some of the widget options which multiple widgets support are described here for brevity. For options that take screen units, values are in pixels unless an optional one letter suffix modifier is present — **c** (cm), **i** (inch), **m** (mm), or **p** (points).

-activebackground *color*

Background color of widget when it is active.

A *color* is any of the valid names for a color defined in the color database, or a specification of the red, green and blue intensities in the form:

#RGB #RRGGBB #RRRGGBBB #RRRRGGGGBBBB

Each *R*, *G*, or *B* represents a single hexadecimal digit. The four forms permit colors to be specified with 4-bit, 8-bit, 12-bit or 16-bit values. When fewer than 16 bits are provided for each color, they represent the most significant bits of the color. For example, #3a7 is the same as #3000a0007000.

-activeborderwidth *width*

Width in screen units of widget border when it is active.

-activeforeground *color*

Foreground color of widget when it is active.

-activestyle *style*

The style in which to draw the active element. One of **dotbox**, **none** or **underline** (default).

-anchor *anchorPos*

How information is positioned inside widget. Valid *anchorPos* values are **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw**, and **center**.

-background *color*

Background color of widget in normal state (Abbrev: **-bg**).

-bitmap *bitmap*

Bitmap to display in the widget (**error**, **gray12**, **gray25**, **gray50**, **gray75**, **hourglass**, **info**, **questhead**, **question**, **warning**, **@filename**).

-borderwidth *width*

Width in screen units of widget border in normal state (Abbrev: **-bd**).

-command *tclCommand*

Tcl command to evaluate when widget is invoked.

-cursor *cursor*

Cursor to display when mouse pointer is in widget. Valid formats:

name [*fgColor* [*bgColor*]

Name of cursor from /usr/include/X11/cursorfont.h.

@sourceName maskName fgColor bgColor

Get source and mask bits from files *sourceName* and *maskName*.

- @sourceName fgColor**
Get source bits from file *sourceName* (background transparent).
- disabledbackground color**
Background color of widget when it is disabled. Use default if *color* is the empty string.
- disabledforeground color**
Foreground color of widget when it is disabled. Use default if *color* is the empty string.
- exportselection boolean**
Whether or not a selection in the widget should also be the X selection.
- font font**
Font to use when drawing text inside the widget.
- foreground color**
Foreground color of widget in normal state (Abbrev: **-fg**).
- height height | textChars**
Height of widget. Units depend on widget.
- highlightbackground color**
Color of the rectangle drawn around the widget when it does not have the input focus.
- highlightcolor color**
Color of the rectangle drawn around the widget when it has the input focus.
- highlightthickness width**
Width in screen units of highlight rectangle drawn around widget when it has the input focus.
- image image**
Image to display in the widget (see Images).
- insertbackground color**
Color to use as background in the area covered by the insertion cursor.
- insertborderwidth width**
Width in screen units of border to draw around the insertion cursor.
- insertofftime milliseconds**
Time the insertion cursor should remain “off” in each blink cycle.
- insertontime milliseconds**
Time the insertion cursor should remain “on” in each blink cycle.
- insertwidth width**
Width in screen units of the insertion cursor.
- jump boolean**
Whether to notify scrollbars and scales connected to the widget to delay updates until mouse button is released.
- justify left | center | right**
How multiple lines line up with each other.
- orient horizontal | vertical**
Which orientation widget should use in layout.
- padx width**
Extra space in screen units to request for the widget in X-direction.
- pady height**
Extra space in screen units to request for the widget in Y-direction.
- readonlybackground color**
Background color to use when the widget *state* is **readonly**. Use default if *color* is the empty string.

- relief flat | groove | raised | ridge | sunken**
3-D effect desired for the widget's border.
- repeatdelay** *milliseconds*
Time a button or key must be held down before it begins to auto-repeat.
- repeatinterval** *milliseconds*
Time between auto-repeats once action has begun.
- selectbackground** *color*
Background color to use when displaying selected items.
- selectborderwidth** *width*
Width in screen units of border to draw around selected items.
- selectforeground** *color*
Foreground color to use when displaying selected items.
- setgrid** *boolean*
Whether this widget controls the resizing grid for its toplevel window.
- state** *state*
Current state of widget: **normal**, **disabled**, **active** (for button-type widgets), or **readonly** (for entry or spinbox widgets.)
- takefocus** *focusType*
If **0** or **1**, signals that the widget should never or always take the focus. If empty, Tk decides. Otherwise, evaluates *focusType* as script with the widget name appended as argument. The return value of the script must be **0**, **1** or empty.
- text** *string*
Text to be displayed inside the widget.
- textvariable** *variable*
Variable which contains a text string to be displayed inside the widget.
- troughcolor** *color*
Color to use for the rectangular trough areas in widget.
- underline** *index*
Integer index of a character to underline in the widget.
- width** *width | textChars*
Width of widget. Units depend on widget.
- wraplength** *length*
Maximum line length in screen units for word-wrapping.
- xscrollcommand** *cmdPrefix*
Prefix for a command used to communicate with horizontal scrollbars.
- yscrollcommand** *cmdPrefix*
Prefix for a command used to communicate with vertical scrollbars.

22. Widget Scroll Commands

The Canvas, Listbox and Text widgets support the following scrolling commands. The Entry and Spinbox widgets support the **xview** command and the **scan** command with the y coordinate dropped.

widget **scan mark** *x y*

Records *x* and *y* as widget's current view anchor.

widget **scan dragto** *x y [gain]*

Shift the view by *gain* (default 10) times the difference between the coordinates *x* and *y* and the current view anchor coordinates.

widget **xview**

Return a two element list specifying the fraction of the horizontal span of the widget at the left and right edges of the window.

widget **xview moveto** *fraction*

Adjust the view in the window so that *fraction* of the total width of the widget is off-screen to the left.

widget **xview scroll** *number* **units** | **pages**

Shift the view by *number* one-tenths (**unit**) or nine-tenths (**pages**) the window's width in the horizontal direction.

widget **yview**

Return a two element list specifying the fraction of the vertical span of the widget at the top and bottom edges of the window.

widget **yview moveto** *fraction*

Adjust the view in the window so that *fraction* of the total height of the widget is off-screen to the top.

widget **yview scroll** *number* **units** | **pages**

Shift the view by *number* one-tenths (**unit**) or nine-tenths (**pages**) the window's height in the vertical direction.

The Text Widget also supports the following:

text **yview** [**-pickplace**] *index*

Changes view of widget's window to make character at *index* visible. If **-pickplace** is specified, *index* will appear at the top of the window.

The Entry (**xview** only), Listbox and Spinbox (**xview** only) Widgets also support the following:

widget **xview** *index*

Adjusts view so that character position *index* is at left edge.

widget **yview** *index*

Adjusts view so that element at *index* is at top of window.

23. Entry Validation

The Entry and Spinbox widgets support entry validation with the use of the **-validate**, **-validatecommand** and **-invalidcommand** options.

The *validateCommand* will be evaluated according to the **-validate** *mode* as follows:

none	No validation will occur.
focus	Evaluate <i>validateCommand</i> when the entry receives or loses focus.
focusin	Evaluate <i>validateCommand</i> when the entry receives focus.
focusout	Evaluate <i>validateCommand</i> when the entry loses focus.
key	Evaluate <i>validateCommand</i> when the entry is edited.
all	Evaluate <i>validateCommand</i> for all above conditions.

It is possible to perform percent substitutions on *validateCommand* and *invalidCommand*, just as you would in a bind script. The following substitutions are recognized:

%d	Type of action: 1 for insert, 0 for delete, or -1 for focus, forced or textvariable validation.
%i	Index of char string to be inserted/deleted if present, otherwise -1.
%P	The value of the entry should edition occur.
%s	The current value of entry before edition.

- %S** The text string being inserted/deleted, if any.
- %v** The type of validation currently set.
- %V** The type of validation that triggered the callback: **key**, **focusin**, **focusout** or **forced**.
- %W** The name of the entry widget.

24. The Canvas Widget

Canvas Options

-background	-insertbackground	-selectborderwidth
-borderwidth	-insertborderwidth	-selectforeground
-cursor	-insertofftime	-takefocus
-height	-insertontime	-width
-highlightbackground	-insertwidth	-xscrollcommand
-highlightcolor	-relief	-yscrollcommand
-highlightthickness	-selectbackground	

-closeenough *float*

How close the mouse cursor must be to an item before it is considered to be “inside” the item.

-confine *boolean*

Whether it is allowable to set the canvas’s view outside the scroll region.

-scrollregion *corners*

List of four coordinates describing the left, top, right, and bottom of a rectangular scrolling region.

-xscrollincrement *distance*

Specifies the increment for horizontal scrolling in screen units.

-yscrollincrement *distance*

Specifies the increment for vertical scrolling in screen units.

Coordinates and distances are specified in screen units which are floating point numbers optionally suffixed with a scale letter. Examples: 5 (pixel), 2.2i (inch), 4.1c (cm), 3m (mm), 21p (pts), $\frac{1}{2}$ inch)

Larger y-coordinates refer to points lower on the screen.

Larger x-coordinates refer to points farther to the right.

Coordinate lists:

The list of coordinates may be specified either as a single Tcl list, or as a sequence of discrete elements.

Tag or id: An *id* is the integer assigned when an item is created while a *tag* is any non-integer form. Tags may be associated with multiple items. A *tagOrId* may be a logical expression using operators '&&', '|', '^', '!' and parenthesized subexpressions.

Tags: **current**

Character indices:

charIndex, **end**, **insert**, **sel.first**, **sel.last**, @*x,y*

Line/polygon indices:

evenNumber, **end**, **insert**, **sel.first**, **sel.last**, @*x,y*

Dash patterns:

intList Each element represents the number of pixels of a line segment. The odd segments are drawn using the **outline** color, the others are transparent.

charString Only the characters [.,-_] are supported. The space can be used to enlarge the space between other line elements. Equivalence of characters to *intList*:

'.'	2 4
'.'	4 4
'-'	6 4
'_'	8 4
'[.,-_]'	Enlarge 2nd number by factor of 2

Canvas Commands

canvas **addtag** *tag searchSpec* [*arg arg ...*]
Add *tag* to the list of tags associated with each item that satisfy *searchSpec*. See Canvas Search Specs below.

canvas **bbox** *tagOrId* [*tagOrId ...*]
Returns a list with four elements giving an approximate bounding box for all the items named by the *tagOrId* arguments.

canvas **bind** *tagOrId* [*sequence* [*command*]]
Associates *command* to be invoked on events specified with *sequence* with the items given by *tagOrId*.

canvas **canvasx** *screenx* [*gridspacing*]
Returns the canvas x-coordinate that is displayed at screen x-coordinate *screenx* possibly rounding to nearest multiple of *gridspacing* units.

canvas **canvasy** *screeny* [*gridspacing*]
Returns the canvas x-coordinate that is displayed at screen y-coordinate *screeny* possibly rounding to nearest multiple of *gridspacing* units.

canvas **coords** *tagOrId* [*x0 y0 ...*]
canvas **coords** *tagOrId* [*coordList*]
Query or modify the coordinates that define an item.

canvas **create** *type x y* [*x y ...*] [*option value ...*]
canvas **create** *type coordList* [*option value ...*]
Create a new item of type *type* at specified coordinates and with list options.

canvas **dchars** *tagOrId first* [*last*]
For items given by *tagOrId*, delete the characters in the range given by *first* and *last* (defaults to *first*), inclusive.

canvas **delete** [*tagOrId ...*]
Delete each of the items given by each *tagOrId*.

canvas **dtag** *tagOrId* [*tagToDelete*]
Remove tag *tagToDelete* from the taglist of items given by *tagOrId*.

canvas **find** *searchSpec* [*arg arg ...*]
Returns a list of the items that satisfy the specification *searchSpec*. See Canvas Search Specs below.

canvas **focus** *tagOrId*
Set the focus to the first textual item given by *tagOrId*.

canvas **gettags** *tagOrId*
Return a list of the tags associated with the first item given by *tagOrId*.

canvas **icursor** *tagOrId index*
Set the insertion cursor for the item(s) given by *tagOrId* to just before the character position *index* .

canvas **index** *tagOrId index*
Returns a decimal string giving the numerical index within *tagOrId* corresponding to character position *index*.

canvas **insert** *tagOrId beforeThis string*
Insert *string* just before character position *beforeThis* in items given by *tagOrId* that support textual insertion.

canvas **itemcget** *tagOrId option*
Returns the value *option* for the item given by *tagOrId*.

canvas **itemconfigure** *tagOrId [option value ...]*
Modifies item-specific options for the items given by *tagOrId*.

canvas **lower** *tagOrId [belowThis]*
Move the items given by *tagOrId* to a new position in the display list just before the first item given by *belowThis*.

canvas **move** *tagOrId xAmount yAmount*
Move the items given by *tagOrId* in the canvas coordinate space by adding *xAmount* and *yAmount* to each items x and y coordinates, respectively.

canvas **postscript** *[option value ...]*
Generate a Encapsulated Postscript representation for part or all of the canvas. See Canvas Postscript Options below.

canvas **raise** *tagOrId [aboveThis]*
Move the items given by *tagOrId* to a new position in the display list just after the first item given by *aboveThis*.

canvas **scale** *tagOrId xOrigin yOrigin xScale yScale*
Re-scale items given by *tagOrId* in canvas coordinate space to change the distance from *xOrigin,yOrigin* by a factor of *xScale,yScale* respectively.

canvas **scan** *args*
See Widget Scroll Commands above.

canvas **select adjust** *tagOrId index*
Adjust nearest end of current selection in *tagOrId* to be at *index* and set the other end to be the new selection anchor.

canvas **select clear**
Clear the selection if it is in the widget.

canvas **select from** *tagOrId index*
Set the selection anchor in *tagOrId* to just before the character at *index*.

canvas **select item**
Return id of the selected item. Returns a empty string if there is none.

canvas **select to** *tagOrId index*
Set the selection to extend between *index* and anchor point in *tagOrId*.

canvas **type** *tagOrId*
Returns the type of the first item given by *tagOrId*.

canvas **xview | yview** *args*
See Widget Scroll Commands above.

Canvas Search Specifications

above *tagOrId*
Selects the item just after the one given by *tagOrId* in the display list.

all Selects all the items in the canvas.

below *tagOrId*
Selects the item just before the one given by *tagOrId* in the display list.

closest *x y [halo] [start]*
Select the topmost, closest item to @*x,y* that is below *start* in the display list.
Any item closer than *halo* to the point is considered to overlap it.

enclosed *x1 y1 x2 y2*
Selects all the items completely enclosed within *x1 y1 x2 y2*.

overlapping *x1 y1 x2 y2*
Selects all the items that overlap or are enclosed within *x1 y1 x2 y2*.

withtag *tagOrId*
Selects all the items given by *tagOrId*.

Common Item Options

-dash *pattern*
Dash pattern for the normal state of an item. Default is a solid line.

-activedash *pattern*
Dash pattern for the active state of an item. Default is a solid line.

-disableddash *pattern*
Dash pattern for the disabled state of an item. Default is a solid line.

-dashoffset *offset*
Starting offset in pixels into the pattern provided by the **-dash** option.

-fill *color*
Color used to fill item's area in its normal state.

-activefill *color*
Color used to fill item's area in its active state.

-disabledfill *color*
Color used to fill item's area in its disabled state.

-outline *color*
Color used to draw the item's outline in its normal state.

-activeoutline *color*
Color used to draw the item's outline in its active state.

-disabledoutline *color*
Color used to draw the item's outline in its disabled state.

-offset *offset*
The stipple offset of the form *x,y* or side where side can be **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw**, or **center**.

-outlinestipple *bitmap*
Stipple pattern used to draw the item's outline in its normal state.

-activeoutlinestipple *bitmap*
Stipple pattern used to draw the item's outline in its active state.

-disabledoutlinestipple *bitmap*
Stipple pattern used to draw the item's outline in its disabled state.

-stipple *bitmap*
Stipple pattern used to fill the the item in its normal state.

-activestipple *bitmap*
Stipple pattern used to fill the the item in its active state.

-disabledstipple *bitmap*
Stipple pattern used to fill the the item in its disabled state.

- state** *state*
Override the canvas widget's global state option.
- tags** *tagList*
Replace any existing tags with *tagList* which may be empty.
- width** *outlineWidth*
Width of the outline drawn around the item's region in its normal state.
- activewidth** *outlineWidth*
Width of the outline drawn around the item's region in its active state.
- disabledwidth** *outlineWidth*
Width of the outline drawn around the item's region in its disabled state.

Canvas Item Types

canvas **create arc** *x1 y1 x2 y2 [option value ...]*

canvas **create arc** *coordList [option value ...]*

- | | |
|--|---|
| -activedash <i>pattern</i> | -disabledoutline <i>color</i> |
| -activefill <i>color</i> | -disabledstipple <i>bitmap</i> |
| -activeoutlinestipple <i>bitmap</i> | -disabledwidth <i>outlineWidth</i> |
| -activeoutline <i>color</i> | -fill <i>color</i> |
| -activestipple <i>bitmap</i> | -offset <i>offset</i> |
| -activewidth <i>outlineWidth</i> | -outlinestipple <i>bitmap</i> |
| -dashoffset <i>offset</i> | -outline <i>color</i> |
| -dash <i>pattern</i> | -state <i>state</i> |
| -disableddash <i>pattern</i> | -stipple <i>bitmap</i> |
| -disabledfill <i>color</i> | -tags <i>tagList</i> |
| -disabledoutlinestipple <i>bitmap</i> | -width <i>outlineWidth</i> |

- extent** *degrees*
Size of the angular range occupied by arc measured counter-clockwise from the start.

- start** *degrees*
Starting angle measured from 3-o'clock position.

- style** *pieslice | chord | arc*
How to "complete" the region of the arc.

canvas **create bitmap** *x y [option value ...]*

canvas **create bitmap** *coordList [option value ...]*

- | | |
|----------------------------|-----------------------------|
| -state <i>state</i> | -tags <i>taglist</i> |
|----------------------------|-----------------------------|
- anchor** *anchorPos*
How to position the bitmap relative to the positioning point for the item: **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw**, or **center** (default).
 - background** *color*
Color to use for the bitmap's '0' valued pixels in its normal state.
 - activebackground** *bitmap*
Color to use for the bitmap's '0' valued pixels in its active state.
 - disabledbackground** *bitmap*
Color to use for the bitmap's '0' valued pixels in its disabled state.
 - bitmap** *bitmap*
Bitmap to display in the item in its normal state.
 - activebitmap** *bitmap*
Bitmap to display in the item in its active state.

-disabledbitmap *bitmap*
 Bitmap to display in the item in its disabled state.

-foreground *color*
 Color to use for the bitmap's '1' valued pixels in its normal state.

-activeforeground *bitmap*
 Color to use for the bitmap's '1' valued pixels in its active state.

-disabledforeground *bitmap*
 Color to use for the bitmap's '1' valued pixels in its disabled state.

canvas **create image** *x y [option value ...]*
canvas **create image** *coordList [option value ...]*

-state *state* **-tags** *taglist*

-anchor *anchorPos*
 How to position the image relative to the positioning point for the item: **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw**, or **center** (default).

-image *name*
 Name of the image to display in its normal state.

-activeimage *name*
 Name of the image to display in its active state.

-disabledimage *name*
 Name of the image to display in its disabled state.

canvas **create line** *x1 y1 ... xN yN [option value ...]*
canvas **create line** *coordList [option value ...]*

-activedash *pattern* **-disabledstipple** *bitmap*
-activefill *color* **-disabledwidth** *outlineWidth*
-activestipple *bitmap* **-fill** *color*
-activewidth *outlineWidth* **-state** *state*
-dashoffset *offset* **-stipple** *bitmap*
-dash *pattern* **-tags** *tagList*
-disableddash *pattern* **-width** *outlineWidth*
-disabledfill *color*

-arrow **none** | **first** | **last** | **both**
 Specify on which ends of the line to draw arrows.

-arrowshape *shape*
 Three element list which describes shape of arrow.

-capstyle **butt** | **projecting** | **round**
 How to draw caps at endpoints of the line. One of **butt** (default), **projecting** or **round**.

-joinstyle **bevel** | **miter** | **round**
 How joints are to be drawn at vertices. One of **bevel**, **miter** (default), or **round**.

-smooth *smoothMethod*
 Should the line be drawn as a set of parabolic splines (**true** or **false**), or as *smoothMethod*? The only built-in method is **bezier**.

-splinesteps *number*
 Degree of smoothness desired for curves.

canvas **create oval** *x1 y1 x2 y2 [option value ...]*
canvas **create oval** *coordList [option value ...]*

-activedash *pattern* **-disabledoutline** *color*

-activefill <i>color</i>	-disabledstipple <i>bitmap</i>
-activeoutlinestipple <i>bitmap</i>	-disabledwidth <i>outlineWidth</i>
-activeoutline <i>color</i>	-fill <i>color</i>
-activestipple <i>bitmap</i>	-offset <i>offset</i>
-activewidth <i>outlineWidth</i>	-outlinestipple <i>bitmap</i>
-dashoffset <i>offset</i>	-outline <i>color</i>
-dash <i>pattern</i>	-state <i>state</i>
-disableddash <i>pattern</i>	-stipple <i>bitmap</i>
-disabledfill <i>color</i>	-tags <i>tagList</i>
-disabledoutlinestipple <i>bitmap</i>	-width <i>outlineWidth</i>

canvas **create polygon** *x1 y1 ... xN yN* [*option value ...*]
canvas **create polygon** *coordList* [*option value ...*]

-activedash <i>pattern</i>	-disabledoutline <i>color</i>
-activefill <i>color</i>	-disabledstipple <i>bitmap</i>
-activeoutlinestipple <i>bitmap</i>	-disabledwidth <i>outlineWidth</i>
-activeoutline <i>color</i>	-fill <i>color</i>
-activestipple <i>bitmap</i>	-offset <i>offset</i>
-activewidth <i>outlineWidth</i>	-outlinestipple <i>bitmap</i>
-dashoffset <i>offset</i>	-outline <i>color</i>
-dash <i>pattern</i>	-state <i>state</i>
-disableddash <i>pattern</i>	-stipple <i>bitmap</i>
-disabledfill <i>color</i>	-tags <i>tagList</i>
-disabledoutlinestipple <i>bitmap</i>	-width <i>outlineWidth</i>

-joinstyle *style*

How joints are to be drawn at vertices. One of **bevel**, **miter** (default), or **round**.

-smooth *boolean*

Should the polygon be drawn as a set of parabolic splines.

-splinesteps *number*

Degree of smoothness desired for curved perimeter.

canvas **create rectangle** *x1 y1 x2 y2* [*option value ...*]
canvas **create rectangle** *coordList* [*option value ...*]

-activedash <i>pattern</i>	-disabledoutline <i>color</i>
-activefill <i>color</i>	-disabledstipple <i>bitmap</i>
-activeoutlinestipple <i>bitmap</i>	-disabledwidth <i>outlineWidth</i>
-activeoutline <i>color</i>	-fill <i>color</i>
-activestipple <i>bitmap</i>	-offset <i>offset</i>
-activewidth <i>outlineWidth</i>	-outlinestipple <i>bitmap</i>
-dashoffset <i>offset</i>	-outline <i>color</i>
-dash <i>pattern</i>	-state <i>state</i>
-disableddash <i>pattern</i>	-stipple <i>bitmap</i>
-disabledfill <i>color</i>	-tags <i>tagList</i>
-disabledoutlinestipple <i>bitmap</i>	-width <i>outlineWidth</i>

canvas **create text** *x y* [*option value ...*]
canvas **create text** *coordList* [*option value ...*]

-activefill <i>color</i>	-fill <i>color</i>
-activestipple <i>bitmap</i>	-state <i>state</i>
-disabledfill <i>color</i>	-stipple <i>bitmap</i>
-disabledstipple <i>bitmap</i>	-tags <i>tagList</i>

-pagex *position*

Set the x-coordinate of the positioning point on the page to *position*.

-pagey *position*

Set the y-coordinate of the positioning point on the page to *position*.

-rotate *boolean*

Whether the printed area is to be rotated 90 degrees. (“landscape”).

-width *size*

Specifies the width of the area of the canvas to print. Defaults to the width of the canvas window

-x *position*

Set the x-coordinate of the left edge of canvas area to print.

-y *position*

Set the y-coordinate of the top edge of canvas area to print.

Tkspline 0.4 Package



<http://www.graphviz.org/pub/Tkspline0.4.tar.gz>

An additional smoothing method for canvas line and polygon items. Package command is:

package require Tkspline

canvas **create line ... -smooth spline ...**

canvas **create polygon ... -smooth spline ...**

Spline smoothing requires $3n + 1$ points, where n is the number of spline segments.

The curves generated with the standard **-smooth true** option have the following properties:

- the curve is always tangential to a straight line between consecutive points.
- the curve is only guaranteed to intersect the first and last points of lines.
- the curve is not guaranteed to intersect any points of polygons.

With **-smooth spline** the curves generated have the following different properties:

- the curve is guaranteed to intersect the first point, and every third point after that.
- each segment of the curve shares endpoints with the adjacent segments, but is otherwise independent of them.
- the curve is guaranteed to be tangential to a line between n and $n + 1$ at point n , and also to a line between $n + 2$ and $n + 3$ at point $n + 3$.
- the curve is not guaranteed to be smooth at the junctions between segments unless the shared point and the points either side of it are on a straight line.

25. The Entry Widget

Entry Widget Options

-background

-borderwidth

-cursor

-disabledbackground

-highlightcolor

-highlightthickness

-insertbackground

-insertborderwidth

-relief

-selectbackground

-selectborderwidth

-selectforeground

-disabledforeground	-insertofftime	-state
-exportselection	-insertontime	-takefocus
-font	-insertwidth	-textvariable
-foreground	-justify	-width
-highlightbackground	-readonlybackground	-xscrollcommand

-invalidcommand *tclCommand*

Script to evaluate when *validateCommand* returns false. (Abbrev: **-invcmd**).

-show *char*

Show *char* rather than actual characters for each character in entry.

-validate *mode*

Mode in which validation should operate: **none** (default), **focus**, **focusin**, **focusout**, **key**, or **all**. See Validation above.

-validatecommand *tclCommand*

Evaluate *tclCommand*, which must return a boolean, to validate the input into the entry widget. If it returns false then the addition is rejected and will not occur, and the *invalidCommand* will be evaluated. (Abbrev: **-vcmd**).

Entry Indices: *number*, **anchor**, **end**, **insert**, **sel.first**, **sel.last**, *@x-coord*

Entry Widget Commands

entry **bbox** *index*

Returns bounding box of character given by *index*.

entry **delete** *first* [*last*]

Delete characters from *first* through character just before *last*.

entry **get**

Returns the entry's string.

entry **icursor** *index*

Display insertion cursor just before character at *index*.

entry **index** *index*

Returns the numerical index corresponding to *index*.

entry **insert** *index string*

Insert *string* just before character at *index*.

entry **scan** *option args*

See Widget Scroll Commands above.

entry **selection adjust** *index*

Adjust nearest end of current selection to be at *index* and set the other end to the anchor point.

entry **selection clear**

Clear the selection if currently in the widget.

entry **selection from** *index*

Set the anchor point to be at *index*.

entry **selection present**

Returns 1 if any characters are selected, 0 otherwise.

entry **selection range** *start end*

Select the characters from *start* through character just before *end*.

entry **selection to** *index*

Set the selection to extend between *index* and anchor point.

entry **validate**

Force an evaluation of *validateCommand*.

entry **xview** *args*

See Widget Scroll Commands above.

26. The Listbox Widget

Listbox Widget Options

-activestyle	-height	-selectforeground
-background	-highlightbackground	-setgrid
-borderwidth	-highlightcolor	-state
-cursor	-highlightthickness	-takefocus
-disabledforeground	-relief	-width
-exportselection	-selectbackground	-xscrollcommand
-font	-selectborderwidth	-yscrollcommand
-foreground		

-listvariable Name of a variable, the value of which is a list to be displayed inside the widget.

-selectmode *mode* One of **single**, **browse** (default), **multiple** or **extended**

Listbox Indices: *number* (starts at 0), **active**, **anchor**, **end**, @*x,y*

Listbox Item Options

-background	-selectbackground	-selectforeground
-foreground		

Listbox Widget Commands

listbox **activate** *index*

Sets the active element to *index*.

listbox **bbox** *index*

Returns a list {*x y width height*} bounding element at *index*.

listbox **curselection**

Returns list of indices of all elements currently selected.

listbox **delete** *index1* [*index2*]

Delete range of elements from *index1* to *index2* (defaults to *index1*).

listbox **get** *index1* [*index2*]

Return as a list contents of elements from *index1* to *index2*.

listbox **index** *index*

Returns position *index* in *number* notation.

listbox **insert** *index* [*element* ...]

Insert specified elements just before element at *index*.

listbox **itemcget** *index* *option*

Returns the value *option* for the item given by *index*.

listbox **itemconfigure** *index* [*option value*] ...

Query or Modify item-specific options for the items given by *option*.

listbox **nearest** *y*

Return index of element nearest to *y*-coordinate.

listbox **scan** *args*

See Widget Scroll Commands above.

listbox **see** *index*

Adjust the view in window so element at *index* is completely visible.

listbox **selection anchor** *index*

Set the selection anchor to element at *index*.

listbox **selection clear** *first* [*last*]

De-select elements between *first* and *last* inclusive.

listbox **selection includes** *index*

Returns 1 if element at *index* is selected, 0 otherwise.

listbox **selection set** *first* [*last*]

Add all elements between *first* and *last* inclusive to selection.

listbox **size**

Returns number of elements in listbox.

listbox **xview** | **yview** *args*

See Widget Scroll Commands above.

27. The Menu Widget

Menu Widget Options

-activebackground	-borderwidth	-foreground
-activeborderwidth	-cursor	-relief
-activeforeground	-disabledforeground	-takefocus
-background	-font	

-postcommand *tclCommand*

Specify Tcl command to invoke immediately before the menu is posted.

-selectcolor *color*

Specifies indicator color for checkbox and radiobutton entries.

-tearoff *boolean*

Whether to include a tear-off entry at top of menu.

-tearoffcommand *tclCmd*

Specifies command to be run when menu is torn off. The name of the menu and the new torn-off window will be appended on invocation.

-title *string*

Uses *string* for title of window used when the menu is torn off.

-type *type*

Used at creation where *type* is one of **menubar**, **tearoff**, or **normal**.

Entry Types: **cascade**, **checkboxbutton**, **command**, **radiobutton**,
separator

Menu Indices: *number*, **end**, **active**, **last**, **none**, *@y-coord*,
matchPattern

Menu Widget Commands

menu **activate** *index*

Change state of entry at *index* to be sole active entry in menu.

menu **add** *type* [*option value* ...]

Add new entry of type *type* to bottom of menu. See below for options.

menu clone *newMenuName* [*cloneType*]
 Clones menu as a new menu *newMenuName* of type *cloneType* (see **-type**).

menu delete *index1* [*index2*]
 Delete all entries between *index1* and *index2* inclusive.

menu entrycget *index option*
 Return current value of *option* for entry at *index*.

menu entryconfigure *index* [*option value ...*]
 Set option values for entry at *index*.

menu index *index*
 Returns the numerical index corresponding to *index*.

menu insert *index type* [*option value ...*]
 Same as **add** but inserts new entry just before entry at *index*.

menu invoke *index*
 Invoke the action of the menu entry at *index*.

menu post *x y*
 Display menu on screen at root-window coordinates given by *x y*.

menu postcascade *index*
 Post sub-menu associated with cascade entry at *index*.

menu type *index*
 Returns type of menu entry at *index*.

menu unpost
 Unmap window so it is no longer displayed.

menu yposition *index*
 Returns the y-coordinate within the menu window of the topmost pixel in the entry specified by *index*.

Menu Entry Options

The following options work for all cascade, checkbutton, command, and radiobutton entries unless otherwise specified.

-activebackground	-bitmap	-image
-activeforeground	-font	-state
-background	-foreground	-underline

-accelerator *string*
 Specifies string to display at right side of menu entry.

-columnbreak *value*
 When *value* is 1, entry appears at top of a new column in menu.

-command *tclCommand*
 TCL command to evaluate when entry is invoked.

-compound *how*
 Should both image and text be displayed, and if so, how the image is placed relative to the text. The value *how* is one of **bottom**, **center**, **left**, **none** (default), **right** or **top**.

-hidemargin *value*
 When *value* is 1, the standard margins are not drawn around entry.

-indicatoron *boolean*
 Whether indicator for checkbutton or radiobutton entry should be displayed.

-label *string*
 Textual string to display on left side of menu entry.

- menu** *pathName*
Pathname to a menu to post when cascade entry is active.
- offvalue** *value*
Value to store in checkbox entry's associated variable when de-selected.
- onvalue** *value*
Value to store in checkbox entry's associated variable when selected.
- selectcolor** *color*
Color for indicator in checkbox and radiobutton entries.
- selectimage** *image*
Image to draw in indicator for checkbox and radiobutton entries.
- value** *value*
Value to store in radiobutton entry's associated variable when selected.
- variable** *variable*
Name of global variable to set when checkbox or radiobutton is selected.

28. The Text Widget

Text Widget Options

- | | | |
|-----------------------------|----------------------------|---------------------------|
| -background | -highlightthickness | -selectbackground |
| -borderwidth | -insertbackground | -selectborderwidth |
| -cursor | -insertborderwidth | -selectforeground |
| -exportselection | -insertofftime | -setgrid |
| -font | -insertontime | -state |
| -foreground | -insertwidth | -takefocus |
| -height | -padx | -width |
| -highlightbackground | -pady | -xscrollcommand |
| -highlightcolor | -relief | -yscrollcommand |
-
- autoseparators** *boolean* Should separators be automatically inserted in the undo stack.
 - maxundo** *integer* Maximum compound undo actions.
 - spacing1** *size* Space in screen units above paragraphs.
 - spacing2** *size* Space in screen units between paragraph lines.
 - spacing3** *size* Space in screen units below paragraphs.
 - tabs** *tabList*
Set of tab stops as a list of screen distances giving their positions. Each stop may be followed by one of **left**, **right**, **center**, or **numeric**.
 - undo** *boolean* Should undo mechanism be enabled.
 - wrap none | char | word** How to wrap lines.

Text Indices

- Syntax: *base [modifier ...]*
- Base: *line.char*, *@x,y*, **end**, *mark*, *tag.first*, *tag.last*, *pathName* (embedded window), *imageName* (embedded image)
- Modifier: \pm *count* **chars**, \pm *count* **lines**, **linestart**, **lineend**, **wordstart**, **wordend**
- Ranges: Ranges include all characters from the start index up to but not including the character at the stop index.

Text Tag Options

-background	-justify	-spacing2
-borderwidth	-relief	-spacing3
-font	-spacing1	-wrap
-foreground		

-bgstipple <i>bitmap</i>	Stipple pattern for background.
-elide <i>boolean</i>	Whether the data should be elided (not displayed and takes no space.)
-fgstipple <i>bitmap</i>	Stipple pattern for foreground.
-lmargin1 <i>size</i>	Left margin of first line of a paragraph.
-lmargin2 <i>size</i>	Left margin of wrapped lines of a paragraph.
-offset <i>size</i>	Offset of baseline from normal baseline.
-overstrike <i>boolean</i>	Whether to overstrike text.
-rmargin <i>size</i>	Right margin of all lines.
-tabs <i>tabList</i>	Set of tab stops (see -tabs above).
-underline <i>boolean</i>	Whether to underline text.

Text Embedded Window Options

-align top center bottom baseline	How window is vertically aligned with its line.
-create <i>tclCommand</i>	Script to create and return window pathname if no -window option is given.
-padx <i>width</i>	Extra space in screen units to leave on the left and right side of window.
-pady <i>height</i>	Extra space in screen units to leave on the top and bottom of window.
-stretch <i>boolean</i>	Whether window should be stretched vertically to fill line.
-window <i>pathName</i>	Name of window to display.

Text Embedded Image Options

-align top center bottom baseline	Where image is displayed on the line.
-image <i>image</i>	Specifies Tk image to use for embedded image.
-name <i>imageName</i>	Specifies name which may be used to reference the embedded image.
-padx <i>width</i>	Extra space in screen units to leave on the left and right side of image.
-pady <i>height</i>	Extra space in screen units to leave on the top and bottom of image.

Text Widget Commands

text **bbox** *index*

Returns a list $\{x\ y\ width\ height\}$ bounding character at *index*.

text **compare** *index1* *op* *index2*

Compares indices *index1* and *index2* according to relational operator *op*.

text **debug** *boolean*

Turn on/off debugging and internal consistency checks in the B-tree code associated with text widgets.

text **delete** *index1* [*index2* ...]

Delete range of given text range.

text **dlineinfo** *index*

Returns a list $\{x\ y\ width\ height\ baseline\}$ describing the screen area taken by display line at *index*.

text **dump** [*switches*] *index1* [*index2*]

Returns detailed info on text widget contents in given text range. Switches include **-all**, **-image**, **-mark**, **-tag**, **-text**, **-window** for specifying type of info returned. The switch **-command** *command* exists to invoke a procedure on each element type in the range.

text **edit modified** [*boolean*]

Query or set the modified flag.

text **edit redo**

Reapply last undone edits provided no other edits were done since then.

text **edit reset**

Clears the undo and redo stacks.

text **edit separator**

Inserts a separator (boundary) on the undo stack.

text **edit undo**

Undoes the last edit action. An edit action is defined as all the insert and delete commands that are recorded on the undo stack between two separators.

text **get** *index1* [*index2* ...]

Returns characters in given range. With multiple indices, a list is returned.

text **image cget** *index* *option*

Return current value of *option* for embedded image at *index*.

text **image configure** *index* [*option* [*value* [*option* *value* ...]]]

Modifies embedded image-specific options for the image at *index*.

text **image create** *index* [*option* *value* ...]

Create a new embedded image at position *index* with specified options.

text **image names**

Returns list of names of all images embedded in text widget.

text **index** *index*

Returns position *index* in *line.char* notation.

text **insert** *index* [*string* [*tagList* *string* *tagList* ...]]

Insert *string* into text at *index* applying tags from *tagList*.

text **mark gravity** *markName* [**left** | **right**]

Returns (or sets) which adjacent character a mark is attached to.

text **mark names**

Returns a list of the names of all marks currently set.

text mark next | **previous** *index*
 Return name of next/previous mark at or after/before *index*.

text mark set *markName index*
 Set mark *markName* to position just before character at *index*.

text mark unset *markName [markName ...]*
 Remove each mark specified so they are no longer usable as indices.

text scan *args*
 See Widget Scroll Commands above.

text search [*switches*] *pattern index [stopIndex]*
 Returns index of first character matching *pattern* in text range *index* to *stopIndex*. Switches: **-forwards**, **-backwards**, **-exact**, **-regex**, **-count** *var*, **-nocase**, **-elide**

text see *index*
 Adjust the view in window so character at *index* is completely visible.

text tag add *tagName index1 [index2 ...]*
 Apply tag *tagName* to characters in given range.

text tag bind *tagName [sequence [tclCommand]]*
 Arrange for *tclCommand* to be run whenever event *sequence* occurs for a character with tag *tagName*.

text tag cget *tagName option*
 Return current value of *option* for tag *tagName*.

text tag configure *tagName [option [value [option value ...]]]*
 Modifies tag-specific options for the tag *tagName*.

text tag delete *tagName [tagName ...]*
 Delete all tag information for given tags.

text tag lower *tagName [belowThis]*
 Change priority of tag *tagName* so it is just below tag *belowThis*.

text tag names [*index*]
 Returns a list of the names of all tags associated with character at *index*. If *index* is not given, returns list of all tags defined in widget.

text tag nextrange *tagName index1 [index2]*
 Searches character range *index1* to *index2* (default **end**) for the first region tagged with *tagName*. Returns character range of region found.

text tag prevrange *tagName index1 [index2]*
 Like **nextrange** but searches backwards from *index1* to *index2* (default 1.0).

text tag raise *tagName [aboveThis]*
 Change priority of tag *tagName* so it is just above tag *aboveThis*.

text tag ranges *tagName*
 Returns a list describing all character ranges tagged with *tagName*.

text tag remove *tagName index1 [index2 ...]*
 Remove tag *tagName* for all characters in range *index1* to *index2*.

text window cget *index option*
 Return current value of *option* for embedded window at *index*.

text window configure *index [option [value [option value ...]]]*
 Modifies embedded window-specific options for the window at *index*.

text window create *index [option value ...]*
 Create a new embedded window at position *index* with specified options.

text window names
 Returns list of names of all windows embedded in text widget.

29. Other Standard Widgets

Button

-activebackground	-foreground	-repeatdelay
-activeforeground	-height	-repeatinterval
-anchor	-highlightbackground	-state
-background	-highlightcolor	-takefocus
-bitmap	-highlightthickness	-text
-borderwidth	-image	-textvariable
-command	-justify	-underline
-cursor	-padx	-width
-disabledforeground	-pady	-wraplength
-font	-relief	

-compound *option*

Should the button display both an image and text, and if so, where the image should be placed relative to the text. Valid options are **bottom**, **center**, **left**, **none** (default), **right** and **top**.

-default *state*

Set state of default ring, one of **active**, **normal**, or **disabled**.

-overrelief **flat** | **groove** | **raised** | **ridge** | **sunken**

Alternative relief for the button when the mouse cursor is over the widget.

button **flash**

Alternate checkbutton between active and normal colors.

button **invoke**

Toggle the selection state of the checkbutton and invoke the Tcl command specified with **-command** (if present.)

Checkbutton

-activebackground	-font	-pady
-activeforeground	-foreground	-relief
-anchor	-height	-state
-background	-highlightbackground	-takefocus
-bitmap	-highlightcolor	-text
-borderwidth	-highlightthickness	-textvariable
-command	-image	-underline
-cursor	-justify	-width
-disabledforeground	-padx	-wraplength

-indicatoron *boolean*

Whether or not the indicator should be drawn.

-offrelief **flat** | **raised**

The relief (default **raised**) when the indicator is not drawn and the checkbutton is off.

-offvalue *value*

Value given to variable specified with **-variable** option when the checkbutton is de-selected.

-onvalue *value*
Value given to variable specified with **-variable** option when the checkbutton is selected.

-overrelief **flat** | **groove** | **raised** | **ridge** | **sunken**
Alternative relief for the checkbutton when the mouse cursor is over the widget.

-selectcolor *color*
Color used to fill in indicator when selected.

-selectimage *image*
Image displayed in indicator when selected.

-variable *variable*
Variable to associate with checkbutton.

checkbutton **deselect**
Deselect the checkbutton.

checkbutton **flash**
Alternate checkbutton between active and normal colors.

checkbutton **invoke**
Toggle the selection state of the checkbutton and invoke the Tcl command specified with **-command** (if present.)

checkbutton **select**
Select the checkbutton.

checkbutton **toggle**
Toggle the selection state of the checkbutton.

Frame

-borderwidth	-highlightcolor	-relief
-cursor	-highlightthickness	-takefocus
-height	-padx	-width
-highlightbackground	-pady	

-background *color*
Same as standard **-background** option except it may be the empty string to preserve colormap.

-class *name*
Class name to use in querying the option database and for bindings.

-colormap *colormap*
Color map to use for window. May be the word **new** or pathname of another window.

-container *boolean*
Whether the frame will be a container to embed another application.

-visual *visual*
Visual to use for the window if different from parent.

Label

-activebackground	-foreground	-pady
-activeforeground	-height	-relief
-anchor	-highlightbackground	-takefocus
-background	-highlightcolor	-text
-bitmap	-highlightthickness	-textvariable

-borderwidth	-image	-underline
-cursor	-justify	-width
-disabledforeground	-padx	-wraplength
-font		

-compound *where*

Should the label display both an image and text, and where the image should be placed relative to the text: one of **bottom**, **center**, **left**, **none** (default), **right** or **top**.

Labelframe

-borderwidth	-highlightbackground	-relief
-cursor	-highlightcolor	-takefocus
-font	-highlightthickness	-text
-foreground	-padx	-width
-height	-pady	

-background *color*

Same as standard **-background** option except *color* may be the empty string to preserve colormap.

-class *name*

Class name to use in querying the option database and for bindings.

-colormap *colormap*

Color map to use for window. May be the word **new** or pathname of another window.

-container *boolean*

Whether the labelframe will be a container to embed another application.

-labelanchor *anchorPos*

Where to place the label. Valid *anchorPos* values are **n**, **ne**, **en**, **e**, **es**, **se**, **s**, **sw**, **ws**, **w**, **wn**, and **nw** (default).

-labelwidget *window*

Widget to use as label.

-visual *visual*

Visual to use for the window if different from parent.

Menubutton

-activebackground	-foreground	-relief
-activeforeground	-height	-state
-anchor	-highlightbackground	-takefocus
-background	-highlightcolor	-text
-bitmap	-highlightthickness	-textvariable
-borderwidth	-image	-underline
-cursor	-justify	-width
-disabledforeground	-padx	-wraplength
-font	-pady	

-compound *where*

Should the menubutton display both an image and text, and where the image

should be placed relative to the text: one of **bottom**, **center**, **left**, **none** (default), **right** or **top**.

-direction *direction*

Where to pop up menu where *direction* is one of **above**, **below**, **left**, **right** and **flush**.

-indicatoron *boolean*

If true then a small indicator will be displayed on the button's right side and the default menu bindings will treat this as an option menubutton.

-menu *pathName*

Pathname of menu widget to post when button is invoked.

Message

-anchor	-highlightbackground	-relief
-background	-highlightcolor	-takefocus
-borderwidth	-highlightthickness	-text
-cursor	-justify	-textvariable
-font	-padx	-width
-foreground	-pady	

-aspect *integer*

Ratio of text width to text height times 100 to use to display text.

Panedwindow

-background	-height	-relief
-borderwidth	-orient	-width
-cursor		

-handlepad *offset*

Offset (pixels) from top or left of sash to draw handle.

-handlesize *length*

Side length (pixels) of a sash handle.

-opaqueresize *boolean*

Whether panes should be resized as a sash is moved (true), or deferred until the sash is placed (false).

-sashcursor *cursor*

Cursor to display when over a sash.

-sashpad *padding*

Amount of padding (pixels) to leave on each side of a sash.

-sashrelief **flat** | **groove** | **raised** | **ridge** | **sunken**

Relief to use when drawing a sash.

-sashwidth *width*

Width (pixels) of each sash.

-showhandle *boolean*

Should sash handles be shown.

panedwindow **add** *window* ... [*option value* ...]

Add *window* ... to *panedwindow*, each in a separate pane. The *panedwindow* Management *option value* pairs are described below.

panedwindow **forget** *window* ...

Remove pane containing *window* from the *panedwindow*.

panedwindow **identify** *x y*

Identify the *panedwindow* component underneath the window coordinates *x y*. If over a sash or sash handle, return `{index sash | handle}`, otherwise return an empty list.

panedwindow **proxy coord**

Return list containing *x y* coordinates of most recent sash proxy (used for rubberband-style pane resizing) location.

panedwindow **proxy forget**

Remove proxy (used for rubberband-style pane resizing) from display.

panedwindow **proxy place** *x y*

Place proxy (used for rubberband-style pane resizing) at coordinates *x y*.

panedwindow **sash coord** *index*

Return current *x y* coordinates of top-left corner of region containing sash given by *index* (an integer from 0 to 1 less than the number of panes in *panedwindow*).

panedwindow **sash dragto** *index x y*

Move sash at *index* by difference between *x y* and coordinates given to last sash **mark** command.

panedwindow **sash mark** *index x y*

Records *x y* coordinates for sash at *index*.

panedwindow **sash place** *index x y*

Move sash at *index* to coordinates *x y*.

panedwindow **panecget** *window option*

Query Panedwindow Management *option* (described below).

panedwindow **paneconfigure** *window [option [value [option value ...]]]*

Query or modify Panedwindow Management *option* (described below) for *window*.

panedwindow **panes**

Return ordered list of widgets managed by *panedwindow*.

Panedwindow Management Options

-after *window*

Insert new window after *window*.

-before *window*

Insert new window before *window*.

-height *size*

Specify height (pixels) for window. The *size* may be an empty string.

-minsize *n*

Size of window cannot be made less than *n* (pixels).

-padx *n*

Extra space (pixels) to leave on each side of window.

-pady *n*

Extra space (pixels) to leave on top and bottom of window.

-sticky *style*

For windows smaller than the containing pane, position or stretch the window based on *style* (string containing zero or more of the characters [nsew]).

-width *size*

Specify width (pixels) for window. The *size* may be an empty string.

Radiobutton

-activebackground	-font	-pady
-activeforeground	-foreground	-relief
-anchor	-height	-state
-background	-highlightbackground	-takefocus
-bitmap	-highlightcolor	-text
-borderwidth	-highlightthickness	-textvariable
-command	-image	-underline
-cursor	-justify	-width
-disabledforeground	-padx	-wraplength

-indicatoron *boolean*

Whether or not the indicator should be drawn.

-offrelief **flat** | **raised**

The relief (default **raised**) when the indicator is not drawn and the radiobutton is off.

-overrelief **flat** | **groove** | **raised** | **ridge** | **sunken**

Alternative relief for the checkbutton when the mouse cursor is over the widget.

-selectcolor *color*

Color used to fill in indicator when selected.

-selectimage *image*

Image displayed in indicator when selected.

-value *value*

Value given to variable specified with **-variable** option when the radiobutton is selected.

-variable *variable*

Variable to associate with radiobutton.

radiobutton **deselect**

Deselect the radiobutton.

radiobutton **flash**

Alternate radiobutton between active and normal colors.

radiobutton **invoke**

Toggle the selection state of the radiobutton and evaluate the Tcl command specified with **-command** (if present.)

radiobutton **select**

Select the radiobutton.

Scale

-activebackground	-highlightbackground	-repeatdelay
-background	-highlightcolor	-repeatinterval
-borderwidth	-highlightthickness	-state
-cursor	-orient	-takefocus
-font	-relief	-troughcolor
-foreground		

-bigincrement *number*

A real value to use for large increments of the scale.

-command *tclCommand*
 Specified a TCL command to evaluate when scale's value is changed. The scale's value will be appended as an additional argument.

-digits *integer*
 An integer specifying how many significant digits should be retained.

-from *number*
 A real value corresponding to left or top end of the scale.

-label *string*
 A string to display as label for the scale.

-length *size*
 Specifies the height (width) for vertical (horizontal) scales.

-resolution *number*
 Real value to which scale's value will be rounded to an even multiple of.

-showvalue *boolean*
 Whether or not scale's current value should be displayed in side label.

-sliderlength *size*
 Size of the slider, measured along the slider's long dimension.

-sliderrelief *relief*
 Specify the relief used to display the slider.

-tickinterval *number*
 A real value to specify the spacing between numerical tick marks displayed.

-to *number*
 A real value corresponding to the right or bottom end of the scale.

-variable *variable*
 Name of a global variable to link to the scale.

-width *width*
 Narrow dimension of scale (not including border).

scale coords [*value*]
 Returns x and y coordinates of point corresponding to *value*.

scale get [*x y*]
 If *x y* is given, returns scale value at that coordinate position. Otherwise, scale's current value is returned.

scale identify *x y*
 Returns string indicating part of scale at position *x y*. Maybe one of **slider**, **trough1**, **trough2** or empty.

scale set *value*
 Changes the current value of scale to *value*.

Scrollbar

-activebackground	-highlightcolor	-repeatdelay
-background	-highlightthickness	-repeatinterval
-borderwidth	-jump	-takefocus
-cursor	-orient	-troughcolor
-highlightbackground	-relief	

-activerelief *number*
 Relief to use when displaying the element that is active.

-command *tclCommandPrefix*
Prefix of a Tcl command to evaluate to change the view in the widget associated with the scrollbar.

-elementborderwidth *width*
Width of borders around internal elements (arrows and slider).

-width *width*
Narrow dimension of scrollbar (not including border).

Elements: **arrow1**, **trough1**, **slider**, **trough2**, **arrow2**

scrollbar **activate** *[element]*
Display *element* with active attributes.

scrollbar **delta** *deltaX deltaY*
Returns fractional position change for slider movement of *deltaX deltaY*.

scrollbar **fraction** *x y*
Returns a real number between 0 and 1 indicating where the point given by pixel coords *x y* lies in the trough area of the scrollbar.

scrollbar **get**
Returns current scrollbar settings as the list *{first last}*.

scrollbar **identify** *x y*
Returns name of element under pixel coords *x y*.

scrollbar **set** *first last*
Describes current view of associated widget where *first last* are the percentage distance from widget's beginning of the start and end of the view.

Spinbox

-activebackground	-highlightcolor	-repeatdelay
-background	-highlightthickness	-repeatinterval
-borderwidth	-insertbackground	-selectbackground
-cursor	-insertborderwidth	-selectborderwidth
-disabledbackground	-insertofftime	-selectforeground
-disabledforeground	-insertontime	-state
-exportselection	-insertwidth	-takefocus
-font	-justify	-textvariable
-foreground	-readonlybackground	-width
-height	-relief	-xscrollcommand
-highlightbackground		

-buttonbackground *color*
Background color.

-buttoncursor *cursor*
Cursor to be used when over the spin buttons.

-buttondownrelief *relief*
Relief to be used for the upper spin button.

-buttonuprelief *relief*
Relief to be used for the lower spin button.

-command *tclCommand*
Command to evaluate whenever a spinbutton is invoked. Several percent substitutions are recognized:

%W the widget path

%s the current value of the widget

%d the direction of the button pressed: **up** or **down**

-format *format*
 Alternate format when setting the string value from the **-from** and **-to** range. It must be valid floating-point format specifier of the form **%<pad>.<pad>f**.

-from *float*
 Lowest floating-point value for a spinbox.

-increment *float*
 Floating-point adjustment to value when a spin button is pressed.

-invalidcommand *tclCommand*
 Script to evaluate when *validateCommand* returns false. (Abbrev: **-invcmd**).

-to *float*
 Highest floating-point value for a spinbox.

-validate *mode*
 Mode in which validation should operate: **none** (default), **focus**, **focusin**, **focusout**, **key**, or **all**. See Validation above.

-validatecommand *tclCommand*
 Evaluate *tclCommand*, which must return a boolean, to validate the input into the spinbox widget. If it returns false then the addition is rejected and will not occur, and the *invalidCommand* will be evaluated. (Abbrev: **-vcmd**).

-values *valueList*
 Spinbox contents, starting with the first value.

-wrap *boolean*
 Should the spinbox wrap around the values of data in the widget.

Elements: **buttondown**, **buttonup**, **entry**

Spinbox Indices: *number*, **anchor**, **end**, **insert**, **sel.first**, **sel.last**, *@x-coord*

spinbox **bbox** *index*
 Returns bounding box of character given by *index*.

spinbox **delete** *first* [*last*]
 Delete characters from *first* through character just before *last*.

spinbox **get**
 Returns the spinbox's string.

spinbox **icursor** *index*
 Display insertion cursor just before character at *index*.

spinbox **identify** *x y*
 Returns string (one of **none**, **buttondown**, **buttonup** or **entry**) indicating element of spinbox at position *x y*.

spinbox **index** *index*
 Returns the numerical index corresponding to *index*.

spinbox **insert** *index string*
 Insert *string* just before character at *index*.

spinbox **invoke** *element*
 Invoke the action of the spinbox element **buttondown** or **buttonup**.

spinbox **scan** *option args*
 See Widget Scroll Commands above.

spinbox **selection adjust** *index*

Adjust nearest end of current selection to be at *index* and set the other end to the anchor point.

spinbox **selection clear**

Clear the selection if currently in the widget.

spinbox **selection element** [*element*]

Query or set the currently selected element. If a spinbutton element is specified, it will be displayed depressed.

spinbox **selection from** *index*

Set the anchor point to be at *index*.

spinbox **selection present**

Returns 1 if any characters are selected, 0 otherwise.

spinbox **selection range** *start end*

Select the characters from *start* through character just before *end*.

spinbox **selection to** *index*

Set the selection to extend between *index* and anchor point.

spinbox **set** [*string*]

Query or set spinbox's string.

spinbox **validate**

Force an evaluation of *validateCommand*.

spinbox **xview** *args*

See Widget Scroll Commands above.

Toplevel

-borderwidth

-highlightcolor

-relief

-cursor

-highlightthickness

-takefocus

-height

-padx

-width

-highlightbackground **-pady**

-background *color*

Same as standard but may be empty to preserve colormap space.

-class *string*

Class name for the window to be used by option database.

-colormap *colormap*

Color map to use for window. May be the word **new**, pathname of other toplevel, or empty for the default colormap of screen.

-container *boolean*

Whether toplevel is a container used to embed another application.

-menu *pathName*

Menu widget to be used as a menubar.

-screen *screen*

Screen on which to place the window.

-use *windowID*

Toplevel should be embedded inside window identified by *windowID* (see **winfo id**) which was created as a container.

-visual *visual*

Visual to use for window.

30. Images

image create *type* [*name*] [*options value ...*]

Creates new image of *type* with *options* and returns *name*.

image delete *name*

Deletes the image *name*.

image height *name*

Returns pixel height of image *name*.

image inuse *name*

Returns 1 if the image is in use by any widgets, 0 otherwise.

image names

Returns a list of the names of all existing images.

image type *name*

Returns the type of image *name*.

image types

Returns a list of valid image types.

image width *name*

Returns pixel width of image *name*.

When an image is created, Tk creates a new command with the same name as the image. For all image types, this command supports the **cget** and **configure** methods in the same manner as widgets for changing and querying configuration options.

The bitmap Image Type

-background *color*

Set background color for bitmap.

-data *string*

Specify contents of bitmap in X11 bitmap format.

-file *fileName*

Gives name of file whose contents define the bitmap in X11 bitmap format.

-foreground *color*

Set foreground color for bitmap.

-maskdata *string*

Specify contents of mask in X11 bitmap format.

-maskfile *fileName*

Gives name of file whose contents define the mask in X11 bitmap format.

The photo Image Type

-data *string*

Specify contents of image in a supported format.

-format *formatName*

Specify format for data specified with the **-data** or **-file** options. In standard Tk, the GIF/PGM/PPM formats are supported.

-file *fileName*

Specifies image data should be read from file *fileName*.

-gamma *number*

Gamma correction. Values greater than 1 (default) lighten the image, less than 1 darken the image.

-height *number*
Fixes the height of the image to *number* pixels.

-palette *paletteSpec*
Set the resolution of the color cube to be allocated for image.

-width *number*
Fixes the width of the image to *number* pixels.

imageName **blank**
Blanks the image so has no data and is completely transparent.

imageName **configure** [*option value ...*]
Query or modify the configuration options for the image. If no option is specified, return a list describing all available options. If option is specified with no value, return a list describing the one named option.

imageName **copy** *sourceImage* [*option value ...*]
Copy a region from *sourceImage* to *imageName* using given options.

-from *x1 y1 [x2 y2]*
Specifies rectangular region of source image to be copied.

-to *x1 y1 [x2 y2]*
Specifies rectangular region of target image to be affected.

-shrink
Will clip target image so copied region is in bottom-right corner.

-zoom *x y*
Magnifies source region by *x y* in respective direction.

-subsample *x y*
Reduces source image by using only every *x* *y*th pixel.

-compositingrule *rule*
How transparent pixels are combined. One of **overlay** (default) or **set**.

imageName **data** [*option value ...*]
Return image data. Options are:

-background *color*
If *color* is specified, all transparent pixels will be replaced by *color*.

-format *format-name*
Specifies image format of file.

-from *x1 y1 [x2 y2]*
Specifies a rectangular region of the image file to copy from.

-grayscale
All pixel data will be transformed into grayscale.

imageName **get** *x y*
Returns RGB value of pixel at coords *x y* as list of three integers.

imageName **put** *data* [*options*]
Sets pixels values to *data* (or single color if *data* is a valid color name).
Options are:

-format *format-name*
Specifies image format of *data*.

-from *x1 y1 [x2 y2]*
Specifies a rectangular region of the image data to copy from.

-shrink
Will clip image so copied region is in bottom-right corner.

-to *x y*
Specifies coords of the top-left corner in image to copy into.

imageName **read** *fileName* [*option value ...*]
 Reads image data from file *fileName* (or single color if *fileName* is a valid color name) into image using given options.

- format** *format-name*
 Specifies image format of file.
- from** *x1 y1 x2 y2*
 Specifies a rectangular region of the image file to copy from.
- shrink**
 Will clip image so copied region is in bottom-right corner.
- to** *x y*
 Specifies coords of the top-left corner in image to copy into.

imageName **redither**
 Redither the image.

imageName **transparency get** *x y*
 Returns boolean indicating if the pixel at *x y* is transparent.

imageName **transparency set** *x y boolean*
 Makes the pixel at *x y* transparent if *boolean* is true, opaque otherwise.

imageName **write** *fileName* [*option value ...*]
 Writes image data from image into file *fileName*.

- background** *color*
 If *color* is specified, all transparent pixels will be replaced by *color*.
- format** *format-name*
 Specifies image format for the file.
- from** *x1 y1 x2 y2*
 Specifies a rectangular region of the image to copy from.
- grayscale**
 All pixel data will be transformed into grayscale.

31. Window Information

winfo atom [**-displayof** *window*] *name*
 Returns integer identifier for atom given by *name* on *window*'s display.

winfo atomname [**-displayof** *window*] *id*
 Returns textual name of atom given by integer *id* on *window*'s display.

winfo cells *window*
 Returns number of cells in the colormap for *window*.

winfo children *window*
 Returns list containing path names of *window*'s children in stacking order.

winfo class *window*
 Returns the class name of *window*.

winfo colormapfull *window*
 Return 1 if the colormap for *window* is full, 0 otherwise.

winfo containing [**-displayof** *window*] *rootX rootY*
 Returns path name of window containing the point *rootX rootY* on *window*'s display..

winfo depth *window*
 Returns the depth (bits per pixel) of *window*.

winfo exists *window*
 Returns 1 if *window* exists, 0 if it doesn't.

winfo fpixels *window number*
Returns floating-point value giving the number of pixels in *window* corresponding to the distance given by *number*.

winfo geometry *window*
Returns the pixel geometry for *window*, in the form *widthxheight+x+y*.

winfo height *window*
Returns height of *window* in pixels.

winfo id *window*
Returns a hexadecimal string indicating the platform identifier for *window*.

winfo interps [**-displayof** *window*]
Returns a list of all Tcl interpreters registered on *window*'s display.

winfo ismapped *window*
Returns 1 if *window* is currently mapped, 0 otherwise.

winfo manager *window*
Returns the name of the geometry manager currently responsible for *window*.

winfo name *window*
Returns *window*'s name within its parent, as opposed to its full path name.

winfo parent *window*
Returns the path name of *window*'s parent.

winfo pathname [**-displayof** *window*] *id*
Returns the path name of the window whose X identifier is *id* on *window*'s display.

winfo pixels *window number*
Returns the number of pixels in *window* corresponding to the distance given by *number*, rounded to nearest integer.

winfo pointerx *window*
Returns mouse pointer's x coordinate on *window*'s screen.

winfo pointerxy *window*
Returns mouse pointer's x and y coordinates on *window*'s screen.

winfo pointery *window*
Returns mouse pointer's y coordinate on *window*'s screen.

winfo reqheight *window*
Returns a decimal string giving *window*'s requested height, in pixels.

winfo reqwidth *window*
Returns a decimal string giving *window*'s requested width, in pixels.

winfo rgb *window color*
Returns a list of the three RGB values that correspond to *color* in *window*.

winfo rootx *window*
Returns the x-coordinate, in the root window of the screen, of the upper-left corner of *window* (including its border).

winfo rooty *window*
Returns the y-coordinate, in the root window of the screen, of the upper-left corner of *window* (including its border).

winfo screen *window*
Returns the name of the screen associated with *window*, in the form *displayName.screenIndex*.

winfo screencells *window*
Returns the number of cells in the default color map for *window*'s screen.

winfo screendepth *window*
Returns the depth (bits per pixel) of *window*'s screen.

winfo screenheight *window*
Returns the height in pixels of *window*'s screen.

winfo screenmmheight *window*
Returns the height in millimeters of *window*'s screen.

winfo screenmmwidth *window*
Returns the width in millimeters of *window*'s screen.

winfo screenvisual *window*
Returns the visual class of *window*'s screen. Maybe one of:
directcolor, **grayscale**, **pseudocolor**, **staticcolor**,
staticgray, or **truecolor**.

winfo screenwidth *window*
Returns the width in pixels of *window*'s screen.

winfo server *window*
Returns server information on *window*'s display.

winfo toplevel *window*
Returns the pathname of the top-level window containing *window*.

winfo viewable *window*
Returns 1 if *window* and all its ancestors are mapped, 0 otherwise.

winfo visual *window*
Returns the visual class of *window* (see **winfo screenvisual**).

winfo visualid *window*
Returns the X identifier for the visual of *window*.

winfo visualsavailable *window*
Returns a list whose elements describe the visuals available for *window*'s screen including class and depth..

winfo vrootheight *window*
Returns the height of the virtual root window associated with *window*.

winfo vrootwidth *window*
Returns the width of the virtual root window associated with *window*.

winfo vrootx *window*
Returns the x-offset of the virtual root window associated with *window*.

winfo vrooty *window*
Returns the y-offset of the virtual root window associated with *window*.

winfo width *window*
Returns *window*'s width in pixels.

winfo x *window*
Returns x-coordinate of the upper-left corner of *window* in its parent.

winfo y *window*
Returns y-coordinate of the upper-left corner of *window* in its parent.

32. The Window Manager

wm aspect *window* [*minNumer minDenom maxNumer maxDenom*]
Query, set or cancel *window*'s desired aspect ratio range.

wm attributes *window* [*option* [*value* [*option value* ...]]]
Query or set platform specific attributes for *window*.

wm client *window* [*name*]
Query, set or cancel *window*'s **WM_CLIENT_MACHINE** property which informs window manager of client machine *name* on which the application is running.

wm colormapwindows *window* [*windowList*]
Query or set *window*'s **WM_COLORMAP_WINDOWS** property which identifies *windowList* windows within *window* with private colormaps.

wm command *window* [*value*]
Query, set or cancel *window*'s **WM_COMMAND** property. Informs window manager of command used to invoke the application.

wm deiconify *window*
Arrange for *window* to be mapped on the screen.

wm focusmodel *window* [**active**|**passive**]
Query or set the focus model for *window*.

wm frame *window*
Returns the platform window identifier for the outermost decorative frame containing *window*. If *window* has none, returns platform id of *window* itself.

wm geometry *window* [*newGeometry*]
Query or set geometry of *window*.

wm grid *window*
Return list containing current *baseWidth baseHeight widthInc heightInc* for gridded *window*.

wm grid *window* {} {} {} {}
Indicates *window* is not to be managed as a gridded window.

wm grid *window baseWidth baseHeight widthInc heightInc*
Indicates *window* is to be managed as a gridded window with the specified relation between grid and pixel units.

wm group *window* [*pathName*]
Query, set or cancel *pathName* of group leader for *window*.

wm iconbitmap *window*
Cancel bitmap used as icon image when *window* is iconified. Returns name of previous bitmap.

wm iconbitmap *window bitmap*
Set bitmap to use as icon image when *window* is iconified.

wm iconify *window*
Arrange for *window* to be iconified.

wm iconmask *window*
Cancel bitmap used as mask icon image when *window* is iconified. Returns name of previous bitmap.

wm iconmask *window bitmap*
Set bitmap to use to mask icon image when *window* is iconified.

wm iconname *window* [*newName*]
Query or set name to use as a label for *window*'s icon.

wm iconposition *window* [*x y*]
Query, set or cancel hint for position on root window to place *window*'s icon.

wm iconwindow *window* [*pathName*]
Query, set or cancel *pathName* of window to use as the icon when *window* is iconified.

wm maxsize *window* [*width height*]
Query or set maximum size that *window* may be resized to in each direction.

wm minsize *window* [*width height*]
 Query or set minimum size that *window* may be resized to in each direction.

wm overriddenirect *window* [*boolean*]
 Query or set the override-redirect flag of *window* commonly used by window manager to determine whether window should have a decorative frame.

wm positionfrom *window* [**program** | **user**]
 Indicate from whom the *window*'s current position was requested.

wm protocol *window*
 Return list of all protocols for which *window* has handlers.

wm protocol *window name*
 Return current command associated with *window* for messages of protocol *name*.

wm protocol *window name* {}
 Cancel handler associated with *window* for messages of protocol *name*.

wm protocol *window name command*
 Specify a Tcl command as handler of *name* protocol messages for *window*.

wm resizable *topWindow* [*widthBoolean heightBoolean*]
 Query or set whether *topWindow*'s width and/or height is resizable.

wm sizefrom *window* [**program** | **user**]
 Indicate from whom the *window*'s current size was requested.

wm stackorder *window*
 Return a list of toplevel windows in stacking order, from lowest to highest.

wm stackorder *window1* **isabove** | **isbelow** *window2*
 Return boolean indicating if *window1* is currently above or below *window2* in the stacking order.

wm state *window* [*newstate*]
 Query or set current state of *window*: **normal**, **icon**, **iconic**, or **withdrawn**.

wm title *window* [*string*]
 Query or set *string* as title for *window*'s decorative frame.

wm transient *window*
 Return current master window if *window* is a transient window.

wm transient *window* {}
 Inform window manager that *window* is not a transient window.

wm transient *window* [*master*]
 Inform window manager that *window* is a transient of window *master*.

wm withdraw *window*
 Arranges for *window* to be withdrawn from the screen.

33. Binding and Virtual Events

bind *tag*
 Returns list of all sequences for which a bindings exists for *tag*.

bind *tag sequence*
 Returns the script bound to the given sequence for *tag*.

bind *tag sequence tclCommand*
 Binds *tclCommand* to the given sequence for *tag*. If *tclCommand* is the empty string, the binding is deleted. If the first character of *tclCommand* is a **+**, then it is appended to the currently associated script. Does % substitution on *tclCommand* (See Event Fields below)

bindtags *window* [*tagList*]
 Sets the current precedence order of tags for *window* to *tagList*. If *tagList* is an empty list, the tags are set back to the defaults.

event add <<*virtual*>> *sequence* [*sequence* ...]
 Arrange for virtual event <<*virtual*>> to be triggered when anyone of given *sequences* occur.

event delete <<*virtual*>> [*sequence* ...]
 Deletes given *sequences* (or all if none given) from list that triggers the virtual event <<*virtual*>>.

event generate *window* *event* [**-when** *when*] [*option value* ...]
 Generate *event* in *window* as if it came from window system. Possible options are listed in the **Event Field** table below. The **-when** option sets when the event will be processed. Possible values for *when* are:

now process immediately (default)
tail place at end of event queue
head place at beginning of event queue
mark same as **head** but behind previous generated events

event info [<<*virtual*>>]
 Returns list of sequences that trigger virtual event <<*virtual*>>(if not given, returns list of defined virtual events).

The sequence argument is a list of one or more event patterns. An event pattern may be a single ASCII character, a string of the form <*modifier-modifier-type-detail*>, or <<*name*>>(virtual event).

Modifiers:

Alt	Button4, B4	Meta, M	Mod5, M5
Any	Button5, B5	Mod1, M1	Quadruple
Button1, B1	Control	Mod2, M2	Shift
Button2, B2	Double	Mod3, M3	Triple
Button3, B3	Lock	Mod4, M4	

Types:

Activate	Destroy	Map
ButtonPress, Button	Enter	MapRequest
ButtonRelease	Expose	Motion
Circulate	FocusIn	MouseWheel
CirculateRequest	FocusOut	Property
Colormap	Gravity	Reparent
Configure	KeyPress, Key	ResizeRequest
ConfigureRequest	KeyRelease	Unmap
Create	Leave	Visibility
Deactivate		

Details: for buttons, a number 1 – 5
 for keys, a keysym (/usr/include/X11/keysymdef)

Tags: internal window (applies to just that window)
 toplevel window (applies to all its internal windows)
 window class name (applies to all widgets in class)
 all (applies to all windows)

Event Fields:

Generate Option	Code	Valid Events
	%%	Single ‘%’

-above <i>window</i>	%a	Configure
-borderwidth <i>size</i>	%B	§, Configure
-button <i>number</i>	%b	ButtonPress, ButtonRelease
-count <i>number</i>	%c	Expose
-delta <i>number</i>	%D	MouseWheel
-detail <i>detail</i>	%d	†, ConfigureRequest, Focus
-focus <i>boolean</i>	%f	†
-height <i>size</i>	%h	§, Expose, ResizeRequest
	%i	Window field <i>all events</i>
-keycode <i>number</i>	%k %A	KeyPress, KeyRelease
-keysym <i>name</i>	%K %A %N	KeyPress, KeyRelease
-mode <i>notify</i>	%m	†, Focus
-override <i>boolean</i>	%o	Configure, Map, Reparent
-place <i>where</i>	%p	Circulate, CirculateRequest
-root <i>window</i>	%R	†, †
-rootx <i>coord</i>	%X	†, †
-rooty <i>coord</i>	%Y	†, †
-sendevent <i>boolean</i>	%E	<i>all events</i>
-serial <i>number</i>	%#	<i>all events</i>
-state <i>state</i>	%s	†, †, Visibility
-subwindow <i>window</i>	%S	†, †
-time <i>integer</i>	%t	†, †, Property
	%T	Type field <i>all events</i>
-warp <i>boolean</i>		†
-width <i>size</i>	%w	§, Configure, Expose, ResizeRequest
	%W	path name <i>all events</i>
-x <i>coord</i>	%x	†, †, Configure, Expose, Gravity, Reparent
-y <i>coord</i>	%y	†, †, Configure, Expose,, Gravity, Reparent
†		ButtonPress, ButtonRelease, KeyPress, KeyRelease, Motion
‡		Enter, Leave
§		ConfigureRequest, Create

34. Geometry Management

The pack Command

pack [**configure**] *slave* [*slave ...*] [*options*]

Sets how slave windows should be managed by pack geometry master.

- after** *sibling* **-fill** *none | x | y | both*
- anchor** *anchor* **-in** *master*
- before** *sibling* **-side** *top | bottom | left | right*
- expand** *boolean*
- ipadx** *pixels*
How much horizontal internal padding to leave on each side of the slave(s).
- ipady** *pixels*
How much vertical internal padding to leave on on the top and bottom of the slave(s).

-padx *pixels*
How much horizontal external padding to leave on each side of the slave(s).

-padx {*leftPixels rightPixels*}
How much horizontal external padding to leave on the left and right side of the slave(s).

-pady *pixels*
How much vertical external padding to leave on the top and bottom of the slave(s).

-pady {*topPixels bottomPixels*}
How much vertical external padding to leave on the top and bottom of the slave(s).

pack forget *slave* [*slave* ...]
Unmanages the given slave windows.

pack info *slave*
Returns list containing current pack configuration of window *slave*.

pack propagate *master* [*boolean*]
Enables or disables propagation for the window *master*.

pack slaves *master*
Returns lists of slaves in the window *master*.

The place Command

place *window option value* [*option value* ...]
Sets how *window* should be placed inside its master.

place [configure] *window* [*option* [*value* [*option value* ...]]]
Query or modify how *window* should be placed inside its master.

-anchor <i>anchor</i>	-relheight <i>size</i>	-rely <i>location</i>
-height <i>size</i>	-relwidth <i>size</i>	-x <i>location</i>
-in <i>master</i>	-relx <i>location</i>	-y <i>location</i>
-width <i>size</i>	-bordermode inside outside ignore	

-bordermode *mode*
Degree to which borders within master determine placement of slave. One of **inside** (default), **outside** or **ignore**.

place forget *window*
Unmanages *window*.

place info *window*
Returns list containing current place configuration of *window*.

place slaves *window*
Returns lists of slaves in the window *master*.

The grid Command

grid [configure] *slave* | **x** | ^ [*slave* | **x** | ^ ...] [*option value* ...]
Sets how slave windows should be managed by grid geometry master.

-columnspan <i>n</i>	-in <i>other</i>	-row <i>n</i>
-column <i>n</i>	-rowspan <i>n</i>	-sticky [n][s][e][w]

-ipadx *pixels*
How much horizontal internal padding to leave on each side of the slave(s).

-ipady *pixels*
How much vertical internal padding to leave on on the top and bottom of the slave(s).

-padx *pixels*
How much horizontal external padding to leave on each side of the slave(s).

-padx {*leftPixels rightPixels*}
How much horizontal external padding to leave on the left and right side of the slave(s).

-pady *pixels*
How much vertical external padding to leave on the top and bottom of the slave(s).

-pady {*topPixels bottomPixels*}
How much vertical external padding to leave on the top and bottom of the slave(s).

grid bbox *master* [*column row* [*column2 row2*]]
Returns bounding box in pixels of space occupied by whole grid (no args), the cell (2 args), or area spanning between given cells (4 args).

grid columnconfigure *master* *columnList* [*options*]
Set/query column properties of given columns in grid *master*.

-minsize *size* Minimum size of column.

-pad *amount* Padding to add to sides of largest slave.

-uniform *tag* Groups column with others having same *tag* (an arbitrary string). Apportions space for all in the group in strict proportion to their weights.

-weight *int* Relative weight for apportioning extra space.

grid forget *slave* [*slave ...*]
Removes (and unmaps) each slave from grid forgetting its configuration.

grid info *slave*
Returns list describing configuration state of *slave*.

grid location *master* *x y*
Returns column and row containing screen units *x y* in *master*. If *x y* is outside grid, -1 is returned.

grid propagate *master* [*boolean*]
Set/query whether *master* tries to resize its ancestor windows to fit grid.

grid remove *slave* [*slave ...*]
Removes (and unmaps) each slave from grid remembering its configuration.

grid rowconfigure *master* *rowList* [*options*]
Set/query row properties of given rows in grid *master*. Same options as for **columnconfigure** but for rows.

grid size *master*
Returns size of grid (as *columns rows*) for *master*.

grid slaves *master* [**-row** *row*] [**-column** *column*]
With no options, a list of all slaves in *master* is returned. Otherwise, returns a list of slaves in specified row or column.

Grid Relative Placement

- Increases columnspan of *slave* to the left.

x Leave an empty column.

^ Extends the rowspan of *slave* above.

35. Fonts

font actual *fontDesc* [**-displayof** *window*] [*option*]

Returns actual value for *option* used by *fontDesc* on *window*'s display. If *option* is not given, the complete option/actual value list is returned.

font configure *fontname* [*option* [*value option value* ...]]

Query/set font options for application created font *fontname*.

font create [*fontname* [*option value* ...]]

Create new application font *fontname* with given font options.

font delete *fontname* [*fontname* ...]

Delete given application created fonts.

font families [**-displayof** *window*]

Returns list of know font families defined on *window*'s display.

font measure *fontDesc* [**-displayof** *window*] *text*

Returns width in pixels used by *text* when rendered in *fontDesc* on *window*.

font metrics *fontDesc* [**-displayof** *window*] [*metric*]

Query font metrics of *fontDesc* on *window*'s display where *metric* maybe be one of **-ascent**, **-descent**, **-linespace**, or **-fixed**. If *metric* is not given, the complete metric/value list is returned.

font names

Returns list of application created fonts.

Font Description:

1. *fontname*

Name of font created by the application with **font create**.

2. *systemfont*

Name of platform-specific font interpreted by graphics server.

3. *family* [*size* [*style* ...]]

A Tcl list with first element the name of a font family, the optional second element is desired size, and additional elements chosen from **normal** or **bold**, **roman** or **italic**, **underline** and **overstrike**.

4. X-font name

A Unix-centric font name of the form
-foundry-family-weight-slant-setwidth-addstyle-pixel-point-resx-resy-spacing-width-charset-encoding. The '*' character may be used as a wild card.

5. *option value* [*option value* ...]

A Tcl list of *option/values* as valid for **font create**.

Font Options:

-family <i>name</i>	Font family (e.g. Courier , Times , Helvetica).
-size <i>size</i>	Size in points (or pixels if negative).
-weight <i>weight</i>	Either normal (default) or bold .
-slant <i>slant</i>	Either roman (default) or italic .
-underline <i>boolean</i>	Whether or not font is underlined.
-overstrike <i>boolean</i>	Whether or not font is overstricken.

36. Other Tk Commands

bell [**-displayof** *window*] [**-nice**]

Ring the X bell on *window*'s display. The **-nice** option will attempt to

avoid waking the screen saver.

clipboard clear [**-displayof** *window*]

Claim ownership of clipboard on *window*'s (default **'.'**) display, clearing its contents.

clipboard append [**-displayof** *window*] [**-format** *fmt*] [**-type** *type*] *data*

Append *data* of *type* (default **STRING**) to clipboard on *window*'s (default **'.'**) display.

clipboard get [**-displayof** *win*] [**-type** *type*]

Retrieve the clipboard on *win*'s (default **'.'**) display as *type* (default **STRING**).

destroy [*window window ...*]

Destroy the given windows and their descendents.

focus [**-force**] *window*

Sets the input focus for *window*'s display to *window*. The **-force** option cause the focus to be set even if another application has it.

focus [**-displayof** *window*]

Returns name of focus window on *window*'s display.

focus -lastfor *window*

Returns the window which most recently had focus and is a descendent of *window*'s toplevel .

grab current [*window*]

Returns name of current grab window on *window*'s display. If *window* is omitted, returns list of all windows grabbed by application.

grab release *window*

Releases grab on *window*.

grab [**set**] [**-global**] *window*

Sets a grab on *window* which will be local unless **-global** specified.

grab status *window*

Returns **none**, **local**, or **global** to describe grab state of *window*.

::safe::loadTk *slave* [**-use** *window*] [**-display** *displayName*]

Initialize the required data structures in the safe interpreter *slave* and then load Tk into it.

lower *window* [*belowThis*]

Places *window* below window *belowThis* in stacking order.

option add *pattern value* [*priority*]

Adds option with *pattern value* at *priority* (0 – 100) to database.

option clear

Clears option database and reloads from user's Xdefaults.

option get *window name class*

Obtains option value for *window* under *name* and *class* if present.

option readfile *fileName* [*priority*]

Reads options from Xdefaults-style file into option database at *priority*.

raise *window* [*aboveThis*]

Places *window* above window *aboveThis* in stacking order.

selection clear [**-displayof** *window*] [**-selection** *selection*]

Clears *selection* (default **PRIMARY**) on *window*'s display.

selection get [**-displayof** *window*] [**-selection** *selection*] [**-type** *type*]

Retrieves *selection* from *window*'s display using representation *type*.

selection handle [**-selection** *sel*] [**-type** *type*] [**-format** *fmt*] *win cmd*

Arranges for *cmd* to be run whenever *sel* of *type* is owned by *win*.

selection own [**-displayof** *window*] [**-selection** *selection*]
Returns path name of *window* which owns *selection* on *window*'s display.

selection own [**-selection** *selection*] [**-command** *command*] *window*
Causes *window* to become new owner of *selection* and arranges for *command* to be run when *window* later loses the *selection*.

send [**-displayof** *window*] [**-async**] *interp cmd* [*arg arg ...*]
Evaluate *cmd* with *args* in the Tk application *interp* on *window*'s display. If **-async** is specified, the **send** command will return immediately.

tk appname [*newName*]
Set the interpreter name of the application to *newName*.

tk caret *window* [*option value ...*]
Query or set accessibility caret location for display of *window*. Options are:

- x** *pixels* Window-relative X coordinate.
- y** *pixels* Window-relative Y coordinate.
- height** *pixels* Height of *window* or cursor.

tk scaling [**-displayof** *window*] [*floatNumber*]
Set scaling factor for conversion between physical units and pixels on *window*'s display where *floatNumber* is pixels per point ($\frac{1}{72}$ inch).

tk useinputmethods [**-displayof** *window*] [*boolean*]
Sets and queries whether Tk should use XIM (X Input Methods) for filtering events. The resulting state is returned. Default is true for the main display.

tk windowingsystem
Returns one of **x11**, **win32**, **classic** or **aqua**.

tkwait variable *varName*
Pause program until global variable *varName* is modified.

tkwait visibility *window*
Pause program until *window*'s visibility has changed.

tkwait window *window*
Pause program until *window* is destroyed.

tk_bisque
Set default color palette to old bisque scheme.

tk_chooseColor [*option value ...*]
Pops up dialog for user to choose color and returns choice. Options are:

- initialcolor** *color* Makes default choice *color*.
- parent** *window* Makes *window* parent of dialog.
- title** *string* Makes *string* title of dialog window.

tk_chooseDirectory [*option value ...*]
Pops up dialog for user to select a directory and returns choice. Options are:

- initialdir** *dirname* Makes initial directory *dirname*.
- parent** *window* Makes *window* parent of dialog.
- title** *string* Makes *string* title of dialog window.
- mustexist** *boolean* May non-existent directories be chosen?

tk_dialog *topw title text bitmap default string* [*string ...*]
Pops up dialog using toplevel window *topw* with a button for each *string* argument. Returns index of button user presses, starting from 0 for the leftmost button. The index *default* specifies the default button.

tk_focusNext *window*
Returns the next window after *window* in focus order.

tk_focusPrev *window*
Returns the previous window before *window* in focus order.

tk_focusFollowsMouse
Change focus model of application so focus follows the mouse pointer.

tk_getOpenFile [*option value ...*]
Pops up dialog for user to choose an existing filename and returns choice.
Options are:

- defaultextension** *extension*
String to append to filename if no extensions exists on chosen filename.
- filetypes** *filePatternList*
List of filepattern elements of the form
typeName {extension [extension ...]} [{macType ...}]
- initialdir** *directory*
Display files in *directory*.
- initialfile** *fileName*
Make default choice *fileName*.
- multiple** *boolean*
Allow choice of multiple files.
- parent** *window*
Makes *window* parent of dialog.
- title** *string*
Makes *string* title of dialog window.

tk_getSaveFile [*option value ...*]
Pops up dialog for user to choose a filename and returns choice. Options are same as for **tk_getOpenFile**.

tk_menuSetFocus *menuWindow*
Save the current focus and sets the focus to *menuWindow*. menu name.

tk_messageBox [*option value ...*]
Displays a message dialog and returns the unique symbolic name of button pressed by user. Options are:

- default** *name*
Make button *name* the default.
- message** *string*
Display *string* as dialog's message.
- parent** *window*
Makes *window* parent of dialog.
- title** *string*
Makes *string* title of dialog window.
- icon error | info | question | warning**
Adds specified icon to dialog.
- type** *buttonSet*
One of **abortretryignore**, **ok**, **okcancel**, **retrycancel**, **yesno** or **yesnocancel**.

tk_optionMenu *w varName value [value ...]*
Creates option menu with name *w* consisting of the given values. The current value is stored in global variable *varName*. Returns internal menu name.

tk_popup *menu x y [entry]*
Post popup *menu* so that *entry* is positioned at root coords *x y*.

tk_setPalette *color*
Changes the color scheme for Tk so the default background color is *color* and other default colors are computed.

tk_setPalette *name color [name color ...]*
Set the default color for the named options in the color scheme explicitly.
Possible options are:

activeBackground	highlightColor
activeForeground	insertBackground
background	selectColor
disabledForeground	selectBackground

foreground
highlightBackground

selectForeground
troughColor

tk_textCopy *window*

The default binding for the *copy* key for the text widget.

tk_textCut *window*

The default binding for the *cut* key for the text widget.

tk_textPaste *window*

The default binding for the *paste* key for the text widget.

37. TclX 8.4



<http://sourceforge.net/projects/tclx>

The TclX package extends Tcl's capabilities by adding new commands. Package command is:

```
package require Tclx
```

38. TclX Special Variables and Commands

tclx_library Path to the TclX runtime library.
TCLXENV Array containing information private to TclX.
mainloop The procedure which sets up a top-level event loop.

39. TclX General Commands

dirs List the directories in the directory stack.

commandloop *options*

Create an interactive command loop reading from **stdin** and writing results to **stdout**. In interactive mode, the results of a **set** command with two arguments is not printed.

If **SIGINT** is configured to generate a Tcl error, it can be used to delete the current command being typed without aborting the program in progress.

Options are:

-async

Read from **stdin** until a complete command is available, evaluating it at that point.

-interactive *mode*

Enable or disable interactive command mode. Mode is one of:

on Enable

off Disable

tty Enable if **stdin** is associated with a terminal (default)

-prompt1 *tclCommand*

Use the *result* of evaluating *tclCommand* as the main command prompt, otherwise evaluate **\$tcl_prompt1**.

-prompt2 *tclCommand*

Use the *result* of evaluating *tclCommand* as the continuation command prompt, otherwise evaluate **\$tcl_prompt2**.

-endcommand *tclCommand*

Evaluate *tclCommand* when the command loop terminates.

echo [*string* ...]

Write each *string* (separated by a space) and a final newline to **stdout**.

infox version

Return TclX version number.

infox patchlevel

Return TclX patch level.

infox have_fchown

Return 1 if the **fchown** system call is available, 0 otherwise.

infox have_fchmod
Return 1 if the **fchmod** system call is available, 0 otherwise.

infox have_flock
Return 1 if the **flock** system call is available, 0 otherwise.

infox have_fsync
Return 1 if the **fsync** system call will sync individual files, 0 otherwise.

infox have_ftruncate
Return 1 if the **ftruncate** system call is available, 0 otherwise.

infox have_msgcats
Return 1 if XPG message catalogs are available, 0 if not.

infox have_posix_signals
Return 1 if Posix signals are available, 0 otherwise.

infox have_signal_restart
Return 1 if restartable signals are available, 0 if not.

infox have_truncate
Return 1 if the **truncate** system call is available, 0 otherwise.

infox have_waitpid
Return 1 if the **waitpid** system call is available, 0 otherwise.

infox appname
Return the value of the **C** variable **tclAppName**.

infox applongname
Return the value of the **C** variable **tclLongAppName**.

infox appversion
Return the value of the **C** variable **tclAppVersion**.

infox apppatchlevel
Return the value of the **C** variable **tclAppPatchlevel**.

for_array_keys *varName arrayName tclCommand*
Shortcut for: **foreach** *varName [array names arrayName] tclCommand*

for_recursive_glob *varName dirList globList tclCommand*
Recursively search *dirList* (but do not follow symbolic links) using patterns in *globList*. Evaluate *tclCommand* for each file matched setting *varName* to the name of the file.

loop *varName firstValue limitValue [increment] tclCommand*
Shortcut for: **for** **{set** *varName firstValue*
 { *\$varName compare \$limitValue*
 {incr *varName increment*
 tclCommand

If *increment* (default 1) is negative, the loop counts down. The values of *firstValue*, *limitValue* and *increment* are integer expressions only evaluated once at the beginning of the loop.

popd
Pop the top entry from the directory stack and make it the current directory.

pushd [*dirName*]
Push the current directory onto the directory stack and **cd** to *dirName* (default [**pwd**].)

recursive_glob *dirList globList*
Recursively search *dirList* (but do not follow symbolic links) using patterns in *globList*. Return a list of all files matched.

showproc [*procName* ...]
Show the definition of *procName* (default is all loaded procedures.)

try_eval *tclCommand catchCommand [finalCommand]*

Evaluate *tclCommand* in the current context. If an error occurs and *catchCommand* is not empty, then *catchCommand* is evaluated and its result becomes the result of **try_eval**. The context of *catchCommand* includes the global variables:

errorResult The result, including the error message, of *tclCommand*

errorCode As set by Tcl

errorInfo As set by Tcl

If the error is to be continued, use the following command

error \$errorResult \$errorCode \$errorInfo

If *finalCommand* is not empty, it is evaluated after *tclCommand* and *catchCommand*. If an error occurs within *finalCommand*, then it becomes the result of the **try_eval** command.

40. TclX Debugging Commands

cmdtrace *level options*

cmdtrace on *options*

Trace commands executed depth below or at *level*, or all commands for **on**.

Options are:

noeval Print arguments unevaluated, otherwise evaluated.

notruncate Do not truncate trace lines to 60 characters.

procs Trace procedure calls only.

fileId Write trace output to *fileId* rather than **stdout**.

command *tclProc*

For each line traced, call *tclProc* with arguments:

command The command before any argument substitution.

argv Final argument list.

evalLevel Call level.

procLevel Procedure call level.

Tracing is turned off during *tclProc* execution. The values of **errorInfo** and **errorCode** are saved and restored on return from *tclProc*.

cmdtrace off

Disable tracing.

cmdtrace depth

Returns the current maximum trace level, or zero if trace is disabled.

41. TclX Development Commands

edprocs [*procName ...*]

Write *procName* (default all defined procedures) to a temporary file, invoke the editor specified by **\$env(EDITOR)** (default **vi**), and then source the file if it was changed.

profile [**-commands**] [**-eval**] **on**

Collect performance data by procedure name. If **-commands** is specified, include data for Tcl commands as well. For **-eval**, the call stack rather than the procedure scope stack is used to group statistics.

profile off *profDataVar*

Terminate profiling and store the results in *profDataVar*.

profrep *profDataVar sortKey [fileId] [title]*

Generates a report from *profDataVar* collected by the **profile** command writing to *fileId* (default **stdout**) with optional *title*. The *sortKey* (one of **calls**, **cpu** or **real**) indicates how to sort the data.

saveprocs *fileName [procName ...]*

Saves *procName* (if omitted, all defined procedures) to *fileName*.

42. TclX Unix Access Commands

alarm *float*

Schedule a **SIGALRM** to be signaled after *float* seconds have elapsed. If *float* is zero, cancel any previous request. Returns the number of seconds left in the previous alarm.

execl [**-argv0** *argv0*] *progName [argList]*

Do an *execl*, replacing the current program, with *progName* passing arguments *argList*. The **-argv0** option specifies an alternate value for *argv[0]*.

chroot *dirName*

Invoke the *chroot*(2) system call.

fork

Fork the current Tcl process, returning zero to the child and the child's process number to the parent.

id user [*name*]

id userid [*uid*]

Query or set the real and effective user ID.

id convert userid *uid*

id convert user *name*

Convert user ID number to a user name, or vice versa.

id group [*name*]

id groupid [*gid*]

Query or set the real and effective group ID.

id groups

id groupids

Return the current group names or ID numbers.

id convert groupid *gid*

id convert group *name*

Convert group ID number to a group name, or vice versa.

id effective user

id effective userid

Return the effective user name or ID number.

id effective group

id effective groupid

Return the effective group name or ID number.

id host

Same as **info hostname**.

id process

Same as **pid**.

id process parent

Return the parent process ID of the current process.

id process group

Return the group ID of the current process.

id process group set

Set the process group ID of the current process to its process ID.

kill [-**pgroup**] [*signal*] *pidList*

Send *signal* (default **SIGTERM**) to the each process in *pidList* or process group for **-pgroup**. If present, *signal* is either the signal number or the symbolic name.

link [-**sym**] *srcPath destPath*

Same as **file link**.

nice [*priorityincr*]

Query or set the process priority. Return the current priority.

readdir [-**hidden**] *dirPath*

Returns a list containing the contents (except for "." and "..") of directory *dirPath*.

signal [-**restart**] *action sigList [tclCommand]*

Specify action to take when Unix signal *sigList* (numbers, symbolic names or * for all signals) is received. Use **-restart** to restart blocking system calls if *action* is not error. Action is one of:

default Perform system default action.

ignore Ignore the signal.

error Generate a catchable Tcl error with **\$errorCode** set to **SIGNAME**.

trap Evaluate *tclCommand* and continue if an error is not returned. Percent substitution is done for **%S** (the signal name).

get Return current settings of *sigList* as a keyed list of signal names and values, where each value is a list:

action **default, ignore, error** or **trap**

0 or 1 0 if not blocked, 1 if blocked

tclCommand If action is **trap**

flag indicates if **-restart** is set

set Set signals from a keyed list in the format returned by **get**.

block Block the specified signals.

unblock Unblock the specified signals.

sleep *integer*

The process will sleep for *integer* seconds.

system *command1 [command2 ...]*

Concatenates commands with a space, then evaluate the result using the standard system shell. The exit code of the command is returned.

sync [*fileId*]

Invoke the *sync*(2), or *fsync*(2) if *fileId* is specified, system call.

times

Return a list of the process and child execution times (milliseconds):

utime stime cutime cstime

umask [*octalmask*]

Set or query the file-creation mask.

wait [-**nohang**] [-**untraced**] [-**pgroup**] [*pid*]

Wait for termination of any (or specific *pid*) process created with **execl**. For **-nohang**, don't block but return an empty list if no process has terminated.

For **-untraced**, then the status of stopped children whose status has not yet been reported are also returned. If **-pgroup** is specified and *pid* is not, then wait on any process whose process group ID is they same as the calling process. If both **-pgroup** and *pid* are specified, interpret *pid* as a process group ID.

Return a three-element list:

{ <i>pid</i> EXIT <i>exitCode</i> }	The process exited normally
{ <i>pid</i> SIG <i>signalName</i> }	The process terminated due to a signal
{ <i>pid</i> STOP <i>signalName</i> }	The process is currently stopped

43. TclX File Commands

bsearch *fileId* *key* [*varName*] [*cmpProc*]

Search *fileId*, an ASCII file sorted in ascending order, for a match using *key* and the first white-space delimited field on a line. If *varName* is omitted, return the matched line. Otherwise return 1 for a match (0 if no match) with the line stored in *varName*.

If *cmpProc* is specified, evaluate it with arguments *key* and *line*. The procedure must return -1, 0 or 1 depending on the comparison.

chmod [**-fileid**] *mode* *fileList*

Set permissions of files (or fileIds for **-fileid**) specified in *fileList* to *mode*. Mode can be either numeric or symbolic.

chown [**-fileid**] *idList* *fileList*

Set owner of files (or fileIds for **-fileid**) as specified in *fileList*. The list *idList* contains one or two elements: a user name or number and group name, number or **{ }** (indicating the login group). If no second element, group ownership is unaltered.

chgrp [**-fileid**] *group* *fileList*

Set group ownership of files (or fileIds for **-fileid**) specified in *fileList*. The *group* is a group name or number.

dup *fileId* [*targetFileId*]

Duplicate *fileId* (or a number) returning a new *fileId*. If *targetFileId* is specified, the original file is closed.

fcntl *fileId* *attribute* [*value*]

Query or set file options (boolean) for *fileId*. Attributes are:

RDONLY	opened for input (query only)
WRONLY	opened for output (query only)
RDWR	opened for input and output (query only)
READ	readable (query only)
WRITE	writable (query only)
APPEND	opened for append-only output
NONBLOCK	same as fconfigure -blocking
CLOEXEC	close on exec flag
NOBUF	same as fconfigure -buffering
LINEBUF	same as fconfigure -buffering
KEEPALIVE	keep a socket connection alive.

flock *switches* *fileId* [*start*] [*length*] [*origin*]

Lock all (or part from *start* (default 0) for *length* relative to *origin*) of *fileId*.

If *length* is omitted or 0, lock extends to the end of the file. Origin is one of **start** (default), **current** or **end**. Switches are:

-read create a read lock

-write create a write lock

-nowait return 1 if the lock is obtained, 0 otherwise

for_file *varName filename tclCommand*

Loop over lines in *filename* evaluating *tclCommand* with *varName* set to each line of the file.

funlock *fileId [start] [length] [origin]*

Remove all (or part from *start* (default 0) for *length* relative to *origin*) of a lock placed on *fileId* with **flock**.

fstat *fileId stat [arrayVar]*

Same as **file stat** with an additional item key **tty**. Its value is 1 if *fileId* is associated with a terminal and 0 otherwise. If *arrayVar* is omitted, return a key-value list.

fstat *fileId [item]*

Return individual (identified by *item*) result of stat on *fileId*. Item identifiers are the same as above.

fstat *fileId localhost*

For a socket connection, return a list containing the local IP address, hostname and port number.

fstat *fileId remotehost*

For a socket connection, return a list containing the remote IP address, hostname and port number.

ftruncate [**-fileid**] *file newsize*

Truncate files (or fileIds for **-fileid**) specified in *fileList* to *newsize*.

lgets *fileId [varName]*

Read a Tcl list from *fileId* discarding the final newline. If *varName* is omitted, return the list. Otherwise return the number of characters read and store the list in *varName*.

pipe [*readVar writeVar*]

Create a pipe either setting *readVar* and *writeVar* to the read and write *fileIds*, or return a two-element list of the same.

read_file [**-nonewline**] *fileName*

Read entire contents of *fileName* optionally discarding the last newline character.

read_file *fileName numBytes*

Read at least *numBytes* characters from *fileName*.

select *readFileIds [writeFileIds] [exceptFileIds] [float]*

Wait *float* seconds (default forever) using the *select(2)* system call for zero or more files to become ready for reading, writing, or a pending exception condition. The list of *fileId*'s may be empty. An empty list is returned if the timeout expired, or a three-element list each element of which is a list of the appropriate *fileId*'s.

write_file *fileName string [string ...]*

Write each *string* as a newline terminated line to *fileName*.

44. TclX Network Programming Support

host_info **addresses** *host*
host_info **official_name** *host*
host_info **aliases** *host*

Query the default nameserver for *host* (a name or IP number.)

45. TclX File Scanning Commands

File scanning requires a scan context to search ASCII files. A scan context contains one or more match statements, each of which associate regular expressions with code to be executed when the expressions are matched.

scancontext create

Return a new contextHandle.

scancontext delete *contexthandle*

Delete *contextHandle*.

scancontext copyfile *contexthandle* [*copyFileId*]

Query or set the copyfile *copyFileId* for unmatched lines.

scanfile [**-copyfile** *copyFileId*] *contextHandle* *fileId*

Scan each line (starting from the current file position) from *fileId* using *contextHandle*. If **-copyfile** is specified, *copyFileId* is used for the duration of the command to write all lines unmatched by any pattern or the default pattern.

scanmatch [**-nocase**] *contextHandle* [*regex*] *tclCommand*

Associate *tclCommand* with *regex* in *contextHandle*. If *regex* is omitted, the default match is associated with *tclCommand*. When scanning, a match is attempted with each *regex* (in the order added) and, if successful, *tclCommand* is evaluated. A **continue** command will terminate scanning for the current line, while **return** terminates the **scanmatch** command itself.

The array **matchInfo** is available to *tclCommand* with keys:

line	The matched line.
offset	The file offset of the first character of the matched line.
linenum	The line number (relative to the first line scanned) of the matched line.
context	The current context handle.
handle	The <i>fileId</i> for the file being scanned.
copyHandle	The <i>fileId</i> specified by -copyfile .
submatchN	N^{th} parenthesized sub-expression of the regex.
subindexN	List of starting and ending indices for the N^{th} parenthesized sub-expression of the regex.

46. TclX Math Commands

The following math functions operate as procedures taking the same arguments as the **expr** command. The result is returned.

abs	ceil	floor	log10	sqrt
acos	cos	fmod	pow	tan
asin	cosh	hypot	round	tanh
atan	double	int	sin	
atan2	exp	log	sinh	

Two additional functions are available:

max *num1* [... *numN*]

expr max(*num1*, *num2*)

Return numeric maximum.

min *num1* [... *numN*]

expr min(*num1*, *num2*)

Return numeric minimum.

random *limit*

Return a pseudorandom integer such that $0 \leq \textit{number} < \textit{limit}$.

random seed [*integer*]

Seed the random number generator, optionally providing *integer* seed.

47. TclX List Manipulation Commands

intersect *lista listb*

Return the sorted logical intersection of *lista* and *listb*.

intersect3 *lista listb*

Return a list of three sorted lists: everything in *lista* not in *listb*, the intersection of *lista* and *listb*, and everything in *listb* not in *lista*.

lassign *list varName1* [*varName2* ...]

Assign successive elements of *list* to the specified variables. If *list* contains fewer elements than there are *varNames*, then the additional variables are set to **{}**. If *list* contains more elements than there are *varNames*, then return the unassigned elements.

lcontain *list element*

Return 1 if *element* exists in *list*, 0 otherwise.

lempty *list*

Return 1 if *list* is empty, 0 otherwise.

lmatch [*switches*] *list pattern*

Same as **lsearch -all**. Return all elements of *list* matching *pattern*.

Switches are:

-exact string match

-glob glob pattern match (default)

-regexp regex match

lrmdups *list*

Same as **lsort -unique**.

lvarcat *varName string* [*string* ...]

Form a single list by concatenating each *string* to *varName*. Return and assign the result to *varName*.

lvarpop *varName* [*indexExpr*] [*string*]

Replace or delete (if *string* not present) element *indexExpr* (default 0) of list stored in *varName* with *string*. Return the original value of the element replaced. If *indexExpr* begins with **end** or **len**, it is replaced by the index of the last element or the length of the list respectively.

lvarpush *varName string* [*indexExpr*]

Insert *string* before element *indexExpr* (default 0) of list stored in *varName*. If *indexExpr* begins with **end** or **len**, it is replaced by the index of the last element or the length of the list respectively.

union *lista listb*

Return the sorted union (duplicates removed) of *lista* and *listb*.

48. TclX Keyed Lists

A *keyed list* is a list, each element of which is a list containing a key-value pair. The key-value pairs are referred to as fields. Fields may contain subfields; '.' is the separator character. Subfields are actually fields where the value is another keyed list.

keyldelete *keylistVar key*

Delete the field *key* from the keyed list in *keylistVar*.

keylget *keylistVar [key] [varName]*

Return (or set *varName* with) the value associated with *key* from the keyed list in *keylistVar*. If *varName* is specified, return 1 if *key* was found, 0 otherwise. If *key* is omitted, then return a list of all keys in the keyed list *keylistVar*.

keylkeys *keylistVar [key]*

Return a list of all keys (or subfield keys of item *key*) in the keyed list in *keylistVar*.

keylset *keylistVar key value [key2 value2 ...]*

Set element *key* in the keyed list stored in *keylistVar*, to value.

49. TclX String/Character Commands

ccollate [**-local**] *string1 string2*

Same as **string compare** if **-local** omitted. Otherwise comparison uses the current locale. This command will not work with UTF or binary data.

cconcat [*string1*] [*string2 ...*]

Return the concatenation of the arguments.

cequal *string string*

Same as **string equal**.

cindex *string indexExpr*

Returns the character indexed by the *indexExpr* from *string*. If *indexExpr* begins with **end** or **len**, it is replaced by the index of the last character or the length of the string respectively.

clength *string*

Same as **string length**.

crange *string firstExpr lastExpr*

Return characters from *string* indexed by *firstExpr* through *lastExpr*. If either *firstExpr* or *lastExpr* begins with **end** or **len**, it is replaced by the index of the last character or the length of the string respectively.

csubstr *string firstExpr lengthExpr*

Return *lengthExpr* characters from *string* indexed by *firstExpr*. If either *firstExpr* or *lastExpr* begins with **end** or **len**, it is replaced by the index of the last character or the length of the string respectively.

ctoken *stringVar separators*

Return the first string delimited by *separators* from the string stored in *stringVar*. Replace the contents of *stringVar* with the remainder of its original value.

ctype **[*-failindex* varName]** *class string*

Return 1 if all characters in *string* are of *class*, 0 otherwise. If **-failindex** is specified, store the index of the first non-matching character in *varName*. Classes are:

alnum	characters are alphabetic or numeric
alpha	characters are alphabetic
ascii	characters are ASCII
cntrl	characters are control characters
digit	characters are decimal digits
graph	characters are printable but not white-space
lower	characters are lowercase
space	characters are white-space
print	characters are printable
punct	characters are punctuation
upper	characters are uppercase
xdigit	characters are hexadecimal digits

ctype char *number*

Return the Unicode character equivalent to *number*.

ctype ord *character*

Return the decimal Unicode value of *character*.

replicate *string integer*

Same as **string repeat**.

translit *from to string*

Translate characters in *string*, replacing *from* characters with the corresponding *to* characters. Both *from* and *to* may contain character ranges in the form 'X-Y'. This command only works with ascii characters.

50. TclX XPG/3 Message Catalog Commands

catopen **[*-fail*]** *catName*

catopen **[*-nofail*]** *catName*

Return *catHandle*, the result of opening message catalog *catName*. If **-fail** (default **-nofail**) is specified, an error occurs if the open fails.

catgets *catHandle setNumber msgNumber defaultMessage*

Retrieve message *msgNumber* in *setNumber* from *catHandle*. If not found, return *defaultMsg*.

catclose **[*-fail*]** *catHandle*

catclose **[*-nofail*]** *catHandle*

Close *catHandle*. If **-fail** (default **-nofail**) is specified, an error occurs if the close fails.

51. *Img* 1.2.4 Package



<http://members1.chello.nl/~j.nijtmans/img1.2.4.tar.gz>

This package adds the **pixmap** image type, and provides the additional photo image types **bmp**, **png**, **jpeg**, **tiff**, **xbm**, **xpm** and **window**. Package command is:

package require Img

The pixmap Image Type

-data *string*

Specify contents of pixmap in XPM format.

-file *fileName*

Gives name of file whose contents define the pixmap in XPM format.

Additional photo Image Types

For the *imageName* **read** and *imageName* **write** commands, the **-format** option is used to provide image type specific options. These options are appended to the format string.

imageName **read** *filename* **-format** "bmp"

imageName **write** *filename* **-format** "bmp"

imageName **read** *filename* **-format** "gif [*option value*]"

-index *n*

Selects a specific image from a multi-image GIF file.

imageName **write** *filename* **-format** "gif"

-background *color*

Usually only valid for the **bitmap** image type, for GIF it now may be used to indicate a transparent color.

imageName **read** *filename* **-format** "jpeg [*option value* ...]"

-fast *boolean*

Fast, low-quality processing.

-grayscale *boolean*

Force incoming image to grayscale

imageName **write** *filename* **-format** "jpeg [*option value* ...]"

-grayscale *boolean*

Create monochrome file.

-quality *n*

Compression quality (0..100; 5 – 95 is useful range). Default is 75.

-smooth *n*

Perform smoothing (10 – 30 is enough for most GIF's). Default is 0.

-optimize *boolean*

Optimize Huffman table.

-progressive *boolean*

Create progressive file.

imageName **read** *filename* **-format** "png"

imageName **write** *filename* **-format** "png [*option value* ...]"

Each *option value* is used to write a text chunk such as Author, Title, Description, etc.

-background *color*

Usually only valid for the **bitmap** image type, for PNG it now may be used to indicate a transparent color.

imageName **read** *filename* **-format** "postscript" [*option value* ...]"

-zoom *x* [*y*]

Multiply image size by given scale factors. If *y* is missing, the default is the same as *x*. *x* and *y* are allowed to be in floating point format, but they are rounded to the nearest practically possible value. For postscript the zoom factors should be multiples of $\frac{1}{72}$.

imageName **read** *filename* **-format** "tiff"

imageName **write** *filename* **-format** "tiff" [*option value* ...]"

-compression *type*

May be one of **deflate**, **jpeg**, **packbits**, **lzw**, or **none**.

-byteorder *which*

May be one of **bigendian**, **littleendian**, **network**, **smallendian** or **{}**.

image create photo ... **-format** **window** **-data** *pathname*

Pathname must be an existing window, and must be currently mapped.

imageName **read** *filename* **-format** "xbm"

imageName **write** *filename* **-format** "xbm"

imageName **read** *filename* **-format** "xpm"

imageName **write** *filename* **-format** "xpm"

52. Tcllib 1.4



<http://tcllib.sourceforge.net/>

The Tcl Standard Library is a collection of Tcl packages that provide useful utility functions.

::math

Utility math functions. Package command is:

package require math

::math::cov *value value* ...

Return the coefficient of variation expressed as percent of two or more numeric values.

::math::fibonacci *n*

Return the *n*'th Fibonacci number.

::math::integrate {*x0 y0 x1 y1 x2 y2 x3 y3 x4 y4* ... }

Return a list containing the area under a *curve* defined by a set of at least five *x,y* pairs and the error bound.

::math::max *value* ...

Return the maximum of one or more numeric values.

::math::mean *value* ...

Return the arithmetic mean, or average of one or more numeric values.

::math::min *value* ...

Return the minimum of one or more numeric values.

::math::prod *value* ...

Return the product of one or more numeric values.

::math::random [*value1* [*value2*]]

Return a random number. With no arguments, return a floating point value between 0 and 1. With one argument, return an integer between 0 and *value1*. With two arguments, return an integer between *value1* and *value2*.

::math::sigma *value value ...*

Return the population standard deviation of two or more numeric values.

::math::stats *value value ...*

Return a list containing the mean, standard deviation, and coefficient of variation expressed as percent of two or more numeric values.

::math::sum *value ...*

Return the sum of one or more numeric values.

::profiler

Provide a simple Tcl source code profiler. It collects only function-level information, not the more detailed line-level information. Profiling is initiated via the **::profiler::init** command. Package command is:

package require profiler

::profiler::init

Initiate profiling. All procedures created after this command is called will be profiled.

::profiler::dump *pattern*

Dump profiling information for the all functions matching *pattern* (default all.) The result is a list of key/value pairs that maps function names to information about that function. The information about each function is in turn a list of key/value pairs:

totalCalls

The total number of times *functionName* was called.

callerDist

A list of key/value pairs mapping each calling function that called *functionName* to the number of times it called *functionName*.

compileTime

The runtime, in clock clicks, of *functionName* the first time that it was called.

totalRuntime

The sum of the runtimes of all calls of *functionName*.

averageRuntime

Average runtime of *functionName*.

descendantTime

Sum of the time spent in descendants of *functionName*.

averageDescendantTime

Average time spent in descendants of *functionName*.

::profiler::print [*pattern*]

Print profiling information for all functions matching *pattern* (default all.)

::profiler::reset [*pattern*]

Reset profiling information for all functions matching *pattern* (default all.)

::profiler::resume [*pattern*]

Resume profiling for all functions matching *pattern* (default all.)

::profiler::sortFunctions *key*

Return list of functions sorted by a particular profiling statistic. Values for

key are: **calls**, **exclusiveTime**, **compileTime**, **nonCompileTime**, **totalRuntime**, **avgExclusiveTime**, and **avgRuntime**. The result is a list of lists, where each sublist consists of a function name and the value of *key* for that function.

::profiler::suspend [*pattern*]

Suspend profiling for all functions matching *pattern* (default all.)

::struct::graph

Create and manipulate directed graph objects. Package command is:

package require struct

::struct::graph *graphName*

Create a new graph object with an associated global Tcl command whose name is *graphName*.

graphName **arc append** *arc* [**-key** *key*] *value*

Append *value* to current value associated with *key* (default **data**) for *arc*.

graphName **arc delete** *arc* ...

Remove the specified *arcs* from the graph.

graphName **arc exists** *arc*

Return true if the specified arc exists in the graph.

graphName **arc get** *arc* [**-key** *key*]

Return the value associated with *key* (default **data**) for *arc*.

graphName **arc getall** *arc*

Return list of all key/value pairs for *arc*.

graphName **arc insert** *start* *end* [*child*]

Insert an arc named *child* (or a generated arc name) beginning at the node *start* and ending at *end*.

graphName **arc keyexists** *arc* [**-key** *key*]

Return true if *key* (default **data**) exists for *arc*.

graphName **arc keys** *arc*

Return list of all keys for *arc*.

graphName **arc lappend** *arc* [**-key** *key*] *value*

Append *value* (as a list) to current value associated with *key* (default **data**) for *arc*.

graphName **arc set** *arc* [**-key** *key*] [*value*]

Set or get the key (default **data**) value associated with *arc*. If *value* omitted, return current value.

graphName **arc source** *arc*

Return the node that *arc* begins at.

graphName **arc target** *arc*

Return the node that *arc* ends at.

graphName **arc unset** *arc* [**-key** *key*]

Remove a keyed (default **data**) value from *arc*.

graphName **arcs** [[**-key** *key*] [**-value** *value*]] [*-connection nodelist*]

Return a list of all arcs. The list can be limited to arcs based on the keyed values associated with the arc, the nodes that are connected by the arc, or both. The *-connection* restriction may be one of

-in All arcs whose target is one of *nodelist*.

-out Return all arcs whose source is one of *nodelist*.

-adj Return all arcs adjacent to at least one of *nodelist*.

-inner Return all arcs adjacent to two of *odelist*.

-embedding Return all arcs adjacent to exactly one of *odelist*.

graphName **destroy**
Destroy the graph.

graphName **get** [**-key** *key*]
Return the value associated with *key* (default **data**) for the graph.

graphName **getall**
Return the value associated with *key* (default **data**) for the graph.

graphName **keyexists** [**-key** *key*]
Return true if *key* (default **data**) exists for the graph.

graphName **keys**
Return list of all keys for the graph.

graphName **node append** *node* [**-key** *key*] *value*
Append *value* to current value associated with *key* (default **data**) for *node*.

graphName **node degree** [**-in** | **-out**] *node*
Return number of (incoming **-in** or outgoing **-out**, default all) arcs adjacent to *node*.

graphName **node delete** *node* ...
Remove *node* and all its arcs from the graph.

graphName **node exists** *node*
Return true if *node* exists in the graph.

graphName **node get** *node* [**-key** *key*]
Return the value associated with *key* (default **data**) for *node*.

graphName **node getall** *node*
Return list of all key/value pairs for *node*.

graphName **node insert** [*child*]
Insert a node named *child* (or a generated name) into the graph. The node has no arcs connected to it. The value "" is assigned to key **data**.

graphName **node keyexists** *node* [**-key** *key*]
Return true if *key* (default **data**) exists for *node*.

graphName **node keys** *node*
Return list of all keys for *node*.

graphName **node lappend** *node* [**-key** *key*] *value*
Append *value* (as a list) to current value associated with *key* (default **data**) for *node*.

graphName **node opposite** *node* *arc*
Return the node at the other end of *arc*, which has to be adjacent to *node*.

graphName **node set** *node* [**-key** *key*] [*value*]
Set or get the key (default **data**) value associated with *arc*. If *value* omitted, return current value.

graphName **node unset** *node* [**-key** *key*]
Remove a keyed (default **data**) value from *node*.

graphName **nodes** [[**-key** *key*] [**-value** *value*]] [-*connection* *odelist*]
Return a list of all nodes. The list can be limited to nodes based on the keyed values associated with the nodes, the nodes that are connected by the nodes, or both. The possible *-connection* restrictions are the same as for method **arcs**.

graphName **set** [**-key** *key*] [*value*]
Set or return one of the keyed values associated with a graph for *key* (default **data**).

graphName **swap** *node1 node2*

Swap the position of *node1* and *node2* in the graph.

graphName **unset** [**-key** *key*]

Remove a keyed value from the graph. If no *key* is specified, **data** is assumed.

graphName **walk** *node* [**-order** *order*] [**-type** *type*] [**-dir** *dir*] **-command** *cmd*

Perform a breadth-first or depth-first walk of the graph starting at *node* going in either the direction of outgoing or opposite to the incoming arcs.

-type bfs breadth-first

-type dfs depth-first (default)

-order pre pre-order (default)

-order post post-order

-order both both-order

-dir backward the direction opposite to the incoming arcs

-dir forward the direction of the outgoing arcs

As the walk progresses, *cmd* will be evaluated at each node, with the mode (**enter** or **leave**) and values *graphName* and the name of the node appended.

::struct::queue

Create and manipulate queue objects.

::struct::queue *queueName*

Create a new queue object with an associated global Tcl command whose name is *queueName*.

queueName **clear**

Remove all items from the queue.

queueName **destroy**

Destroy the queue.

queueName **get** [*count*]

Return and remove the front *count* (default 1) items of the queue. If *count* is 1, the result is a simple string; otherwise it is a list.

queueName **peek** [*count*]

Return the front *count* (default 1) items of the queue. If *count* is 1, the result is a simple string; otherwise it is a list.

queueName **put** *item* ...

Put *item* into the queue.

queueName **size**

Return the number of items in the queue.

::struct::stack

Create and manipulate stack objects.

::struct::stack *stackName*

Create a new stack object with an associated global Tcl command whose name is *stackName*.

stackName **clear**

Remove all items from the stack.

stackName **destroy**

Destroy the stack.

stackName **peek** [*count*]

Return the top *count* (default 1) items of the stack. If count is 1, the result is a simple string; otherwise it is a list.

stackName **pop** [*count*]

Return and remove the top *count* (default 1) items of the stack. If count is 1, the result is a simple string; otherwise it is a list.

stackName **push** *item* ...

Put *item* onto the stack.

stackName **size**

Return the number of items on the stack.

::struct::tree

Create and manipulate tree objects.

::struct::tree *treeName*

Create a new tree object with an associated global Tcl command whose name is *treeName*.

treeName **append** *node* [**-key** *key*] *value*

Appends *value* to one of the keyed values (default **data**) associated with *node*.

treeName **children** *node*

Return a list of the children of *node*.

treeName **cut** *node*

Removes *node* from the tree, but not its children. The children of *node* are made children of the parent of *node*.

treeName **delete** *node* ...

Remove *node* and all its children from the tree.

treeName **depth** *node*

Return number of steps from *node* to the root node.

treeName **destroy**

Destroy the tree.

treeName **exists** *node*

Return true if *node* exists in the tree.

treeName **get** *node* [**-key** *key*]

Return value associated with *key* (default **data**) for *node*.

treeName **getall** *node*

Return list of key/value pairs (suitable for use with **array set** for *node*).

treeName **index** *node*

Returns index of *node* in its parent's list of children.

treeName **insert** *parent* *index* [*child* ...]

Insert *child* ... (default a generated name) in the child list of *parent* at **index**. If *index* is **end**, the new nodes will be added after the current last child. If *parent* is **root**, it refers to the root of the tree. A new node has the value "" assigned to key **data**. If *child* already exist in *treeName*, it will be moved from its original location. Return a list of nodes added.

treeName **isleaf** *node*

Return true if *node* is a leaf of the tree.

treeName **keys** *node*

Return list of keys for *node*.

treeName **keyexists** *node* [-**key** *key*]
 Return true if *key* (default **data**) exists for *node*.

treeName **lappend** *node* [-**key** *key*] *value*
 Append *value* (as a list) to one of the keyed (default **data**) values of *node*.

treeName **move** *parent* *index* *node* ...
 Make the *node* children of *parent*, inserting them into the parent's child list at *index*.

treeName **next** *node*
 Return right sibling of *node*, or the empty string if *node* was the last child of its parent.

treeName **numchildren** *node*
 Return number of immediate children of *node*.

treeName **parent** *node*
 Return parent of *node*.

treeName **previous** *node*
 Return left sibling of *node*, or the empty string if *node* was the first child of its parent.

treeName **set** *node* [-**key** *key*] [*value*]
 Set or get one of the keyed (default **data**) values of *node*. If *value* omitted, return current value.

treeName **size** [*node*]
 Return the number of descendants of *node* (default **root**.)

treeName **splice** *parent* *from* [*to*] [*child*]
 Insert *child* (or a generated name) in the child list of *parent* at *from*. If *parent* is **root**, it refers to the root of the tree. The children of *parent* which are in the range of *from* and *to* (default **end**) are made children of *child*. Return *child*.

treeName **swap** *node1* *node2*
 Swap the position of *node1* and *node2*.

treeName **unset** *node* [-**key** *key*]
 Remove a keyed (default **data**) value from *node*.

treeName **walk** *node* [-**order** *order*] [-**type** *type*] -**command** *cmd*
 Perform a breadth-first or depth-first walk of the tree starting at *node*.

-type bfs breadth-first
-type dfs depth-first (default)
-order in in-order
-order pre pre-order (default)
-order post post-order
-order both both-order

As the walk progresses, *cmd* will be evaluated at each node. Percent substitution will be performed on *cmd* before evaluation. The following substitutions are recognized:

%% Insert the literal % character.
%t Name of the tree object.
%n Name of the current node.
%a Name of the action occurring; one of **enter**, **leave**, or **visit**.

53. Tktable 2.8 Package



<http://tktable.sourceforge.net/>

The Tktable package provides a table widget for Tk. Package command is:

package require Tktable

table *pathName options ...*

Create a new Tcl command whose name is *pathName*, a *table* widget. The *pathName* of the window is returned.

Tktable Options

-anchor	-highlightcolor	-invertselected
-background	-highlightthickness	-justify
-cursor	-insertbackground	-relief
-exportselection	-insertborderwidth	-state
-font	-insertofftime	-takefocus
-foreground	-insertontime	-xscrollcommand
-highlightbackground	-insertwidth	-yscrollcommand

-autoclear *boolean*

Does the first keypress in a cell delete previous text (default **false**).

-bordercursor *cursor*

Show *cursor* (default **crosshair**) when over borders.

-borderwidth *pixels*

A value or list of values indicating the width of the 3-D border for interior table cells. (Abbrev: **-bd**).

-browsecommand *tclCommand*

Evaluate *tclCommand* anytime the active cell changes. (Abbrev: **-browsecmd**). Does %-substitution on *tclCommand* (See Tktable Command Substitution below).

-cache *boolean*

Should (default **off**) an internal cache of the table contents be kept.

-colorigin *integer*

What column index (default 0) is the left-most column in the table.

-cols *integer*

Number of columns (default 10) in the table.

-colseparator *character*

A separator character (default a Tcl list) used when cutting or pasting data in a table.

-colstretchmode *mode*

The stretch mode for columns to fill extra allocated window space, one of:

none Do not (default) stretch columns.

unset Only stretch columns without a specific width set.

all Stretch all columns by the same number of pixels.

last Stretch only the last column.

fill Adjust columns to fit window (only valid for **-rowstretch**). This mode may be removed in the future.

-coltagcommand *tclCommand*

Evaluate *tclCommand colNumber* to determine the tag to be used for a given column. It should return the tag name, or a null string.

- colwidth** *width*
Default column width (default 10). Interpreted as characters when *width* is positive, otherwise as pixels.
- command** *tclCommand*
If **-usecommand** is **true**, evaluate *tclCommand* to retrieve cell values instead of the **-variable** array. Does %-substitution on *tclCommand* (See Tktable Command Substitution below).
- drawmode** *mode*
The table drawing mode, one of:
 - slow** Draw with no flashing, but it is slow for larger tables.
 - compatible** Draw directly to the screen (default) using Tk functions.
 - fast** Draw directly to the screen using X functions. This restricts **-borderwidth** to 0 or 1. It is best for large tables, but can flash and is not 100% Tk compatible.
 - single** As **fast**, but only single pixel lines are drawn.
- flashmode** *boolean*
Should (default **false**) cells flash when their value changes.
- flashtime** *integer*
Time in $\frac{1}{4}$ seconds (default 2) for which a cell should flash when its value changes.
- height** *integer*
The desired height in rows. If zero or less, make just large enough to hold all the rows. The height can be further limited by **-maxheight**.
- invertselected** *boolean*
Should (default **false**) the foreground/background of an item simply be swapped rather than merging the sel tag options when the cell is selected.
- ipadx** *pixels*
The internal offset X padding (default 0) for text in a cell. Only affects one side (depending on **-anchor**).
- ipady** *pixels*
The internal offset Y padding (default 0) for text in a cell. Only affects one side (depending on **-anchor**).
- maxheight** *pixels*
The maximum (default 600) height requested by the window.
- maxwidth** *pixels*
The maximum (default 800) width requested by the window.
- multiline** *boolean*
The default setting for the multiline tag option. Default is **true**.
- padx** *pixels*
The left and right X padding (default 0) for a cell.
- pady** *pixels*
The top and bottom Y padding (default 0) for a cell.
- resizeborders** *type*
What kind of interactive border resizing to allow, one of **row**, **col**, **both** (default) or **none**.
- rowheight** *height*
Default row height (default 1). Interpreted as lines when *height* is positive, otherwise as pixels.

- roworigin** *integer*
What row index (default 0) is the top-most row in the table.
- rows** *integer*
Number of rows (default 10) in the table.
- rowseparator** *character*
A separator character (default a Tcl list) used when cutting or pasting data in a table.
- rowstretchmode** *mode*
The stretch mode for rows to fill extra allocated window space. Same modes as **-colstretchmode**.
- rowtagcommand** *tclCommand*
Evaluate *tclCommand* *rowNumber* to determine the tag to be used for a given row. It should return the tag name, or a null string.
- selectioncommand** *tclCommand*
Evaluate *tclCommand* when the selection is retrieved (ie: evaluating "selection get"). (Abbrev: **-selcmd**). Does %-substitution on *tclCommand* (See Tktable Command Substitution below).
- selectmode** *style*
The style for manipulating the selection. One of **single**, **browse** (default), **multiple** or **extended**.
- selecttitle** *boolean*
Should (default **false**) title cells be allowed in the selection.
- selecttype** *mode*
The type of selection for the table, one of **row**, **col**, **cell** (default), or **both** (meaning **row** and **col**).
- sparsearray** *boolean*
Should an associated Tcl array be kept as a sparse array (default **true**).
- titlecols** *integer*
Number of columns (default 0) to use as a title area.
- titlerows** *integer*
Number of rows (default 0) to use as a title area.
- usecommand** *boolean*
Should the **-command** option be used. Automatically set **true** if **-command** is used, but will reset itself **false** if the command returns an error.
- validate** *boolean*
Should validation occur (default **false**) for the active buffer.
- validatecommand** *tclCommand*
Evaluate *tclCommand*, which must return a boolean, to validate the input into the active cell. If it returns false then the addition is rejected and will not occur. (Abbrev: **-vcmd**). Does %-substitution on *tclCommand* (See Tktable Command Substitution below).
- variable** *varName*
Attach global array variable *varName* to the table. Keys are **row**, **col** for cells, or **active** for the value of the active cell buffer.
- width** *integer*
The desired width in columns. If zero or less, make just large enough to hold all the columns. The width can be further limited by **-maxwidth**.
- wrap** *integer*
The default wrap value for tags. Defaults to **false**.

Tktable Indices

row,col, **active**, **anchor**, **bottomright**, **end**, **origin**, **topleft**, @*x,y*

Tktable Tag Options

-anchor	-font	-justify
-background	-foreground	-relief
-borderwidth	-image	-state

-multiline *boolean* Should text be displayed with newlines on multiple lines.

-showtext *boolean* Should the text be shown over an image.

-wrap *boolean* Should characters wrap in a cell that is not wide enough.

Builtin Tag Names: **active**, **flash**, **sel**, **title**

Tktable Embedded Window Options

-background	-padx	-relief
-borderwidth	-pady	

-sticky *sticky* Stickiness of the window inside the cell, as defined by the **grid** command.

-window *pathName* Name of a window to display.

Tktable Command Substitution

Substitution is performed on the *tclCommand*:

	-selectioncommand	-command	-browsecommand	-validatecommand
%c	maximum number of columns in any row	column		
%C	equivalent to %r, %c			
%i	number of cells	0 for get, 1 for set	current cursor position	
%r	number of rows	row		
%s	default value of selection	empty for get, current value for set	index of last active cell	current value
%S	undefined	undefined	index of the new active cell	potential new value
%W	window pathname			

Tktable Commands

table **activate** *index*

Sets the active cell to *index*.

table **bbox** *first* [*last*]

Return a list {*x y width height* }, the bounding box for *first* [*last*].

table **border mark** *x y* [**row** | **column**]

Record *x y* and the row and/or column border under that point, if any. If **row** or **column** is omitted, return a tuple of both border indices (an empty item means no border). Otherwise, just the specified item is returned.

table **border dragto** *x y*
 Compute the difference between *x y* and *x y* of the last **border mark** command. Adjust the previously marked border by the difference.

table **cget** *option*
 General Tk widget **cget** command.

table **clear cache** [*first* [*last*]]
 Clears all or specified section of the cache.

table **clear sizes** [*first* [*last*]]
 Clears all or the specified row and column areas of specific height/width dimensions.

table **clear tags** [*first* [*last*]]
 Clears all or the specified area of all row, column and cell tags.

table **clear all** [*first* [*last*]]
 Performs all of the above clear functions on all or the specified area.

table **configure** [*option* [*value* [*option value* ...]]]
 General Tk widget **configure** command.

table **curselection** [*value*]
 Query the sorted indices of the currently selected cells, or set all the selected cells to the given value.

table **curvalue** [*value*]
 Query or set the value of the cell being edited (indexed by **active**).

table **delete active** *index1* [*index2*]
 Delete the character after *index1*, or from *index1* to *index2* of the **active** cell. An index is a *number*, **insert** or **end**.

table **delete cols** [*switches* [--]] *col* [*count*]
 Delete *count* (default 1) columns starting at (and including) *col*. If count is negative, delete columns to the left, otherwise to the right. Switches are:
-holddimensions Do not adjust the table column dimension (empty columns may appear).
-holdselection Maintain the selection on the absolute cells values. Default is to clear the selection.
-holdtags Do not move tags or adjust widths.
-holdwindows Do not move embedded windows.
-keeptitles Do not change title area cells.

table **delete rows** [*switches* [--]] *row* [*count*]
 Delete *count* (default 1) rows starting at (and including) *row*. If count is negative delete rows going up, otherwise going down. *Switches* are the same as for column deletion.

table **get** *first* [*last*]
 Return a list of values of the cells specified by *first* and *last*.

table **height**
 Return list describing all rows for which a height has been set.

table **height** *row*
 Return height of *row* in characters (positive number) or pixels (negative).

table **height** *row value row value* ...
 Set each *row* to height *value* in lines (positive number) or pixels (negative). If value is **default**, then *row* uses the default height as specified by **-rowheight**.

table hidden
Returns all hidden cells (those cells covered by a spanning cell).

table hidden index
Return the spanning cell covering *index*, if any.

table hidden index index ...
Return 1 if all *indices* are hidden cells, 0 otherwise.

table icursor [arg]
Query or set the location of the insertion cursor in the active cell. *Arg* is one of: 0 (before the first character), **insert** (current insertion point) or **end** (end of the text).

table index index [row|col]
Return cell coordinate corresponding to *index* in the form *row,col*. If **row** or **col** is specified, then return only that portion.

table insert active index string
Insert *string* at *index* (one of *number*, **insert** or **end**) in the active cell.

table insert cols [switches [--]] col [count]
Inserts *count* (default 1) columns starting at *col*. If *count* is negative, insert before *col*, otherwise insert after. *Switches* are the same as for column deletion.

table insert rows [switches [--]] row [count]
Inserts *count* (default 1) rows starting at *row*. If *count* is negative, insert above *row*, otherwise insert below. *Switches* are the same as for column deletion.

table reread
Reread old contents of the cell back into the editing buffer.

table scan args
See Tk Widget Scroll Commands.

table see index
Adjust the view in window so cell at *index* is visible.

table selection anchor index
Set selection anchor to cell *index*.

table selection clear first [last]
Clear any selection of the cells between *first* and *last*. If *first* is **all**, remove selection from all cells.

table selection includes index
Return 1 if the cell at *index* is currently selected, 0 if not.

table selection set first [last]
Select all of the cells between *first* and *last*.

table set index [value [index value ...]]
Sets the cells at *index* to the associated *value*.

table set col index [valueList [index valueList ...]]
Set cells at *index* and subsequent rows to each value in the associated *valueList*.

table set row index [valueList [index valueList ...]]
Set cells at *index* and subsequent columns to each value in the associated *valueList*.

table spans
Return list {*index span ...* } of all known spans.

table spans index
Return the *span* at *index*, if any.

table spans *index rows,cols [index rows,cols ...]*
 Set span beginning at *index* for the specified number of *rows,cols*. A span of **0,0** unsets any span on that cell.

table tag cell *tagName*
 Return list of cells that use *tagName*.

table tag cell *tagName index ...*
 Apply *tagName* to the specified cells. If *tagName* is {}, reset the cells to the default tag.

table tag cget *tagName option*
 Return current value of *option* for tag *tagName*.

table tag col *tagName*
 Return list of columns that use *tagName*.

table tag col *tagName [col ...]*
 Apply *tagName* to the specified columns. If *tagName* is {}, reset the columns to the default tag.

table tag configure *tagName [option [value [option value ...]]]*
 Modifies tag-specific options for the tag *tagName*.

table tag delete *tagName*
 Delete *tagName*.

table tag exists *tagName*
 Return 1 if *tagName* exists, 0 otherwise.

table tag includes *tagName index*
 Return 1 if *index* has tag *tagName*, 0 otherwise.

table tag lower *tagName [belowThis]*
 Change priority of tag *tagName* so it is just below tag *belowThis*.

table tag names [*pattern*]
 Returns a list of the names of all defined tags matching glob *pattern* (default *).

table tag raise *tagName [aboveThis]*
 Change priority of tag *tagName* so it is just above tag *aboveThis*.

table tag row *tagName*
 Return list of rows that use *tagName*.

table tag row *tagName row ...*
 Apply *tagName* to the specified rows. If *tagName* is {}, reset the rows to the default tag.

table validate *index*
 Force an evaluation via the **-validatecommand** on cell at *index*.

table width
 Return list describing all columns for which a width has been set.

table width *col*
 Return width of *col* in characters (positive number) or pixels (negative).

table width *col value col value ...*
 Set each *col* to width *value* in lines (positive number) or pixels (negative). If *value* is **default**, then *col* uses the default width as specified by **-colwidth**.

table window cget *index option*
 Return current value of *option* for window at *index*.

table window configure *index [option [value [option value ...]]]*
 Modifies embedded window-specific options for the cell at *index*.

table **window delete** *index* ...

Delete embedded window from the table and delete the window.

table **window move** *indexFrom indexTo*

Move embedded window from *indexFrom* to *indexTo*. If a window already exists in cell *indexTo*, it will be deleted.

table **window names** [*pattern*]

Return list of all cells containing embedded windows matching glob *pattern* (default *).

table **xview** | **yview** *args*

See Tk Widget Scroll Commands.

54. Vu 2.1.0 Package



<http://tktable.sourceforge.net/>

This package implements Tk bargraph, piechart and dial widgets, and adds stripchart and bargraph canvas item types. Package command is:

package require vu

Bargraph Widget

vu::bargraph *pathName options* ...

Create a new Tcl command whose name is *pathName*, a *bargraph* widget.

The *pathName* of the window is returned.

Bargraph Options

-background **-font** **-highlightthickness**

-borderwidth **-highlightbackground** **-orient**

-cursor **-highlightcolor** **-relief**

-alabels { { *position string useTick* } ... }

Place *string* and/or a tick (if *useTick* is true) at an arbitrarily position along the top or left length of the bar.

-alabfont *font*

The font for **-alabels** *string*.

-fg *color*

-barcolor *color*

Color (default **red**) for the bargraph bar.

-base *baseVal*

The base (default 0) of the bar. $fromVal \leq baseVal \leq toVal$.

-blabels *LabelList*

Like **-alabels**, but for the right or bottom side of the bar.

-barbackground *color*

Background color for the bar region.

-barborderwidth *units*

Width in pixels (default 2) of a bar border.

-blabfont *font*

The font for **-blabels** *string*.

-digits *integer*

Maximum digits (default 6) after the decimal point for floating-point values.

- from** *fromVal*
Value (default 0) displayed at bottom (or left) of a bargraph. If *fromVal* > *toVal*, the bargraph will grow backwards.
- length** *units*
- height** *units*
Total height in pixels (default 100) of a bar.
- label** *string*
Use *string* as a label (placed on top) for the bargraph.
- mode** *mode*
For *mode* **bargraph**, the bar grows from *baseVal* to the current value. For **slider**, a fixed-width slider travels along the bar with its center point indicating the current value.
- padx** *units*
The text elements of a bargraph are held together by *glue* (default 2 pixels) that controls their relative positions with regard to one another and to the central bar. The **-padx** option loosens the glue and makes the elements expand from each other horizontally.
- pady** *units*
Like **-padx**, but controls the vertical *glue*.
- showrange** *boolean*
If true (default), *fromVal* and *toVal* are labeled at the appropriate ends of the bar.
- showvalue** *boolean*
If true (default), the current value is labeled near the left or bottom end of the bar.
- sliderlength** *units*
In **slider** mode, the length (default 10) in pixels of the slider.
- textcolor** *color*
Color for rendering text.
- tickcolor** *color*
Color (default **blue**) for ticks.
- tickinterval** *number*
The interval (default 20.0) between successive ticks. Zero indicates no ticks.
- ticklength** *units*
Length in pixels (default 4) of a tick mark.
- to** *toVal*
Value (default 100) displayed at top (or right) of a bargraph. If *fromVal* > *toVal*, the bargraph will grow backwards.
- width** *units*
Width in pixels (default 20) of a bar.

Bargraph Commands

bargraph **cget** *option*
General Tk widget option retrieval command.

bargraph **configure** [*option1* *value1* *option2* ...]
General Tk widget configuration command.

bargraph **get**
Return the current value of the bargraph.

bargraph **set** [*value*]
Set the bar to height represented by *value*.

Dial Widget

vu::dial *pathName options ...*

Create a new Tcl command whose name is *pathName*, a *dial* widget. The *pathName* of the window is returned.

Dial Options

-activebackground	-highlightbackground	-repeatdelay
-background	-highlightcolor	-repeatinterval
-borderwidth	-highlightthickness	-state
-cursor	-orient	-takefocus
-font	-relief	-variable
-foreground		

-beginangle *integer*

Angle (degrees) at which the scale of the dial will start. Zero is at top and the angle increases clockwise. To start the scale at the left, use a negative value.

-bigincrement *integer*

Size of the large increments. If 0 (default) use $\frac{1}{10}$ the range of the dial.

-command *tclProcName*

Call *tclProcname* with the value of the dial when it changes.

-constrainvalue *boolean*

If true, the value is constrained between the **-from** and **-to** values and **-resolution**. If false (default), only the displayed value is constrained.

-dialborderwidth *units*

Width (default 3) for the inner dial border width.

-dialcolor *color*

Color (default **-background**) for the inner dial.

-dialrelief *relief*

The relief (default **raised**) for the inner dial.

-digits *integer*

Significant digits to retain when converting the value of the dial to a string. If zero, use the smallest value that guarantees that every possible dial position prints as a different string.

-endangle *integer*

Angle (in degrees) at which the scale will end.

-from *float*

Value corresponding to the counterclockwise-most end of the dial.

-label *string*

Use *string* as the label (placed on top) for the dial.

-minortickinterval *float*

The spacing between minor tick marks (those without a numerical value or tag.) If 0 (default), no minor tick marks will displayed.

-mode *type*

One of **circle** (default), or **ellipse**.

-needlecolor *color*

Color of the needle.

-needletype *type*

One of **arc**, **line** (default), or **triangle**.

-pad *units*

The elements of a dial are held together by *glue* (default 2 pixels) that controls their relative positions with regard to one another and to the central dial. The **-pad** option loosens the glue and makes the elements expand from each other.

-radius *units*

Desired radius of the central dial.

-resolution *float*

The resolution (default 1.0) for the dial. If greater than zero then the value, tick marks and endpoints will be rounded to an even multiple of *float*, otherwise no rounding occurs.

-showtags *which*

Which tags to display at normal tick marks: **value** (default), **label**, **both**, or **none**.

-showvalue *boolean*

If true, display the current value.

-tickcolor *color*

Color for the ticks.

-tickinterval *float*

The spacing between major tick marks (those with a numeric value or tag.) If zero, no tick marks are displayed.

-tickwidth *units*

Width of the line that ticks are displayed with. The default 0, means a simple line.

-to *float*

Value corresponding to the clockwise-most end of the dial. The value may be either less than or greater than **-from**.

Dial Commands

dial **cget** *option*

General Tk widget option retrieval command.

dial **configure** [*option1 value1 option2 ...*]

General Tk widget configuration command.

dial **coords** [*value*]

Return a list $\{x\ y\}$, the coordinates of the point represented by *value* (default current value) along the radius of the dial.

dial **get** [*x y*]

Return the value (default current value) corresponding to pixel coordinates *x,y*.

dial **identify** *x y*

Indicate what part of the dial lies under coordinates *x,y*:

dial *x,y* over the dial

left *x,y* to the left of the dial

right *x,y* to the right of the dial

dial **label** [**-constrain**] [*value* [*string* [*value string ...*]]]

Associate *string* with *value* (must be a major tick value). With

-constrain, *value* will be changed if necessary to a valid tick value.

With no arguments, all tags will be shown. With one value, only the label (if any) for that value will be shown.

dial set [*value*]
Query or set the value of the dial.

Pie Widget

vu::pie *pathName options* ...
Create a new Tcl command whose name is *pathName*, a *piegraph* widget.
The *pathName* of the window is returned.

Pie Options

-background	-foreground	-highlightthickness
-borderwidth	-highlightbackground	-relief
-cursor	-highlightcolor	-takefocus
-font		

-angle *integer*
Viewing angle in degrees ($0^\circ \leq integer \leq 90^\circ$). Zero (default) views the pie as a flat circle, 90° as a flat line.

-label *string*
Use *string* as a title.

-legend *string*
What fields are to be displayed in the legend and in what order. The default is **kvpl**. Field identifiers are:
[*number*] **K** Key box (the colored square)
[*number*] **V** Value (the numerical value)
[*number*] **P** P (the percentage of the whole)
[*number*] **L** Label (the given label of the slice)
If *number* is zero, the field is not shown. The default value is -1.
For **K**, a positive value means display the slice's color in a separate box, negative means use the slice's color as background for the whole text.
For **V**, **P** and **L**, a positive value specifies the maximum field length to display, negative means use a field length equal to the longest required for that field.

-origin *integer*
Rotation of the pie in degrees clockwise.

-precision *number*
Precision (default 2) at which to display slice and percentage values.

-padx *width*
Space (default 2) in screen units to the right the pie.

-pady *height*
Space (default 2) in screen units on the bottom of the pie.

-radius *units*
The radius (default 80) of the pie.

-shadow *units*
Depth (default 0) of the shadow drawn when **-angle** is not 0.

-sliceborder *boolean*
If true (default) draw borders around the slices.

-smallfont *font*
The font used to draw the values of the slices at their edges in the main pie area.

Pie Commands

pie **cget** *option*

General Tk widget option retrieval command.

pie **configure** [*option1 value1 option2 ...*]

General Tk widget configuration command.

pie **delete** *pattern ...*

Delete slices that match the given glob *pattern*.

pie **explode** *name1* [*value1* [*name2 value2 ...*]]

Set the explode values of one or more slices. If only one *name* is given, then the explode value for that slice is returned.

pie **itemcget** *name option*

Returns the current value of the configuration option given by *option* for slice *name*.

pie **itemconfigure** *name* [*option1*] [*value1* [*option2 value2 ...*]]

Query or modify the item-specific options for the items given by *name*.

Options are:

-foreground *color*

Foreground color of the slice.

-label *string*

Label (defaults to *name*) of the slice.

-value *float*

Value of the slice.

-explode *units*

Explode value of the slice, how far separated from the main pie to display this piece.

pie **lower** *name* [*belowThis*]

Lower *name* slice below *belowThis* (default the bottom.)

pie **names** [*pattern ...*]

Return the names of the slices, according to glob pattern (default *.)

pie **order** *name ...*

Move the named slices, in order, to the top.

pie **raise** *name* [*aboveThis*]

Raise *name* above *aboveThis* (default the top.)

pie **swap** *name name*

Swap the named slices.

pie **set** *name1* [*value1 name2 value2 ...*]

Set the values of one or more slices. If only *name1* is given, then return its value.

pie **value**

Return the sum of the slice values.

Barchart Canvas Item

A barchart item displays a set of values in a barchart diagram. Scale lines indicate the current scale value.

canvas **create barchart** *x1 y1 x2 y2* [*option value ...*]

-tags *tagList*

-autocolor *boolean*

If **true**, use a built-in 6-color scheme.

- background** *color*
Background color, or transparent if *color* is the empty string.
- barline** *color*
Color (default **black**) of a line drawn between and on top of each bar. If *color* is empty, no line is drawn.
- fill** *color*
Use *color* for the strip. If *color* is empty, the bar is transparent and a line is drawn using **-scaleline** color.
- outline** *color*
Use *color* for the surrounding item frame, or no frame if *color* is empty.
- scaleline** *color*
If a bar value is greater than **-scalevalue** then scalevalue is increased and a scaleline is drawn. If *color* is empty no scaleline is drawn.
- scalelinestyle** *integer*
The scaleline can be dashed: *integer* dots, *integer* empty, Default is 4.
- scalevalue** *float*
Maximum *Y* value (default 100), before a scaleline is drawn.
- values** *valueList*
List of values. By default, one value is initially allocated.

Stripchart Canvas Item

A stripchart item shows a set of values in a X-Y diagram which is scaled automatically. Scale lines indicate the current scale value.

canvas **create stripchart** *x1 y1 x2 y2* [*option value* ...]

- tags** *tagList*
- background** *color*
Background color, or transparent if *color* is the empty string.
- fill** *color*
Fill the area under the stripline with *color*. If *color* is empty, only a line is drawn using **-stripline** color.
- jumpscroll** *integer*
When no more room exists to insert a value, the strip will be moved left by *integer* (default 5) pixels.
- outline** *color*
Use *color* for the surrounding item frame, or no frame if *color* is empty.
-scaleline black black
- scaleline** *color*
If a value is greater than **-scalevalue** then scalevalue is increased and a scaleline is drawn. If *color* is empty no scaleline is drawn.
- scalelinestyle** *integer*
The scaleline can be dashed: *integer* dots, *integer* empty, Default is 4.
- scalevalue** *float*
Maximum *Y* value (default 100), before a scaleline is drawn.
- stripline** *color*
Use *color* for the stripline, or no stripline if *color* is empty.
- values** *valueList*
List of values to append on the right. A filled stripchart will require a list of length $X - \text{jumpscroll} - 2$.



<http://www.vim.org>

The vim (Vi IMproved) editor may be compiled with a Tcl interface. Ex commands invoke the Tcl interpreter, which then provides access to vim via commands and variables in the **::vim** namespace.

The Tcl commands **exit** and **catch** are replaced by custom versions. The **exit** command terminates the current Tcl script (deleting the Tcl interpreter) and returns to vim – use the **:tcl** command to create a new Tcl interpreter. The **catch** command works as usual except that **exit** is not caught. A non-zero exit code causes the ex-command that invoked the Tcl script to return an error.

Vim Ex-mode Commands

:tcl *tclCommand* ...

Create (if necessary) a Tcl interpreter and evaluate *tclCommand*. Tcl objects will persist from one command to the next.

[*range*] **:tcl** <<[*endmarker*]

tclScript

[*endmarker*]

Create (if necessary) a Tcl interpreter and evaluate *tclScript*. The *endmarker* defaults to a dot (.). This form of the **:tcl** command is mainly useful for including tcl code in Vim scripts.

[*range*] **:tcl**do *tclCommand*

Evaluate *tclCommand* for each line in *range* (default 1,\$) setting Tcl global variables **line** and **lnum**. **line** may be modified.

:tclfile *file*

Same as **:tcl source** *file*.

::vim Special Variables and I/O Streams

::vim::current

An array (updated after **::vim::command** is called) containing *current* objects available in vim. The elements are:

buffer The name of the buffer command for the current buffer.

window The name of the window command for the current window.

::vim::lbase

If it is set to '1' (default), then lines and columns start at 1 within **::vim**, otherwise 0.

::vim::range

An array with three elements, **start**, **begin** and **end**. Each is a line number corresponding to the current range. The values of **begin** and **start** are identical.

vimout

Tcl's **stdout** file ID is mapped to **vimout** and any data written to the stream is displayed in the vim message area.

vimerr

Tcl's **stderr** file ID is mapped to **vimerr** and any data written to the stream is displayed in the vim message area.

::vim Commands

Access to vim buffers and windows is via their name similar to way the a window path name is used to refer an individual widget when using Tk.

::vim::beep

Honk. Does not return a result.

::vim::buffer exists *number*

Check if buffer *number* exists.

::vim::buffer *number*

Create a command for buffer *number* returning its name as the result.

::vim::buffer *list*

Create a command for each buffer number in *list* returning a list of their names as the result.

Most buffer commands take line numbers as arguments. In addition to a number (interpreted based on value of **::vim::lbase**), the value can be one of **top**, **start**, **begin**, **first**, **bottom**, **end** or **last**.

buffer **append** *line string*

Append *string* after *line*.

buffer **command** [**-quiet**] *cmd*

Execute vim ex-mode *command* in *buffers* context. The **-quiet** option suppresses error messages from vim.

buffer **count**

Return number of lines in the buffer.

buffer **delcmd** *tclCommand*

Registers *tclCommand* as a deletion callback for the buffer. The command is executed (in the global scope) just before the buffer is deleted.

buffer **delete** *line* [*line*]

Deletes a single or range of lines from the buffer.

buffer **expr** *expression*

Evaluates *expression* using vim's internal expression evaluator in *buffers* context returning the result as a string.

buffer **get** *line* [*line*]

Return a single or range of lines from the buffer.

buffer **insert** *line string*

Insert *string* before *line*.

buffer **last**

Return the number of the last line in the buffer based on value of **::vim::lbase**.

buffer **mark** *mark*

Return the position of the *mark* as string **row number column number**.

buffer **name**

Return the name of the file in the buffer.

buffer **number**

Return the number of this buffer.

buffer **option** *option* [*value*]

Query or set vim option in *buffers* context.

buffer **set** *line string*

buffer **set** *line line list*

Replace a single or range of lines in the buffer. If *list* contains more elements than there are lines to replace, they are inserted. If *list* contains fewer elements, the remaining lines are deleted.

buffer **windows**

Returning a list of window command names created for each window that displays this buffer.

::vim::command [**-quiet**] *command*

Execute vim ex-mode *command*. The **-quiet** option suppresses error messages from vim.

::vim::expr *expression*

Evaluates *expression* using vim's internal expression evaluator returning the result as a string.

::vim::option *option* [*value*]

Query or set vim *option*.

::vim::window list

Create a window command for each window returning a list of command names as the result.

window **buffer**

Create a command for the windows buffer returning its name as the result.

window **command** [**-quiet**] *command*

Execute vim ex-mode *command* in *windows* context. The **-quiet** option suppresses error messages from vim.

window **cursor**

Return the current cursor position as string **row number column number** interpreted based on value of **::vim::lbase**.

window **cursor** *row col*

Set the current cursor position interpreted based on value of **::vim::lbase**.

window **cursor** *arrayName*

Set cursor position from *arrayName* elements **row** and **column** interpreted based on value of **::vim::lbase**.

window **delcmd** *tclCommand*

Registers *tclCommand* as a deletion callback for the window. The command is executed (in the global scope) just before the window is closed.

window **expr** *expression*

Evaluates *expression* using vim's internal expression evaluator in *windows* context returning the result as a string.

window **height**

window **height** *number*

Query or attempt to set the window's height. Return the resulting height.

window **option** *option* [*value*]

Query or set vim option in *windows* context.

Vimconsole 1.2 Package



<http://www.bodenstab.org/>

This package provides an interactive shell in an *xterm*(1) window for the **::vim** Tcl interpreter. Package command is:

:tcl package require Vimconsole

::vimconsole::console

Start the interactive shell. This command is imported into the global namespace.

exit *returnCode*

Stop the shell and return to vim.

Vimconsole Package Variables

tcl_interactive Set to 1.

tcl_rcFileName Set to `~/vimconsole.rc` and automatically source'd each time a new xterm window is opened.

::vimconsole::ttyname

The name returned by `ttyname(3)` in the xterm window.

::math, 91
::pkg::create, 25
::profiler, 92
::struct, 93
::struct::graph, 93
::struct::queue, 95
::struct::stack, 95
::struct::tree, 96
::vim, 112
::vim::beep, 113
::vim::buffer, 113
::vim::command, 114
::vim::expr, 114
::vim::option, 114
::vim::window, 114

after, 26
alarm, 82
append, 15
array, 14
auto_execok, 26
auto_load, 27
auto_mkindex, 27
auto_reset, 27

bell, 74
bgerror, 27
binary, 15
bind, 69
bindtags, 70
bitmap, 63
break, 9
bsearch, 84
button, 53

canvas, 36
case, 9
catch, 27
catclose, 89
catgets, 89
catopen, 89
ccollate, 88
cconcat, 88
cd, 19
cequal, 88

checkbox, 53
chgrp, 84
chmod, 84
chown, 84
chroot, 82
cindex, 88
clength, 88
clipboard, 75
clock, 19
close, 20
cmdtrace, 81
commandloop, 79
concat, 12
continue, 9
crange, 88
csubstr, 88
ctoken, 88
ctype, 89

destroy, 75
dirs, 79
dup, 84

echo, 79
edprocs, 81
encoding, 27
entry, 44
entry validation, 35
eof, 20
error, 27
eval, 27
event, 70
exec, 19
execl, 82
exit, 9
expr, 27

fblocked, 21
fcntl, 84
fconfigure, 21
fcopy, 21
file, 9
fileevent, 21
flock, 84
flush, 21

focus, 75
font, 74
for, 9
for_array_keys, 80
for_file, 85
for_recursive_glob, 80
foreach, 9
fork, 82
format, 15
frame, 54
fstat, 85
ftruncate, 85
funlock, 85

gets, 22
glob, 20
global, 27
grab, 75
grid, 72

history, 27
host_info, 86

id, 82
if, 9
image, 63
lmg, 90
incr, 28
info, 11
infox, 79
interp, 23
intersect, 87
intersect3, 87

join, 12

keyldel, 88
keylget, 88
keylkeys, 88
keylset, 88
kill, 83

label, 54
labelframe, 55
lappend, 12
lassign, 87

lcontain , 87	pipe , 85	system , 83
lempty , 87	pkg_mkIndex , 25	table , 98
lgets , 85	place , 72	tcl_endOfWord , 28
lindex , 12	popd , 80	tcl_findLibrary , 28
link , 83	proc , 28	tcl_startOfNextWord , 28
linsert , 12	profile , 81	tcl_startOfPreviousWord , 28
list , 13	profrep , 82	tcl_wordBreakAfter , 28
listbox , 46	pushd , 80	tcl_wordBreakBefore , 28
llength , 13	puts , 22	tcllib , 91
lmatch , 87	pwd , 20	tclsh , 5
load , 28	radiobutton , 58	TclX , 79
loadTk , 75	raise , 75	tell , 23
loop , 80	random , 87	text , 49
lower , 75	read , 22	time , 28
lrange , 13	read_file , 85	times , 83
lreplace , 13	readdir , 83	tk , 76
lrmdups , 87	recursive_glob , 80	tk_bisque , 76
lsearch , 13	regexp , 15	tk_chooseColor , 76
lset , 13	regsub , 16	tk_chooseDirectory , 76
lsort , 13	rename , 28	tk_dialog , 76
lvarcat , 87	replicate , 89	tk_focusFollowsMouse , 77
lvarpop , 87	return , 9	tk_focusNext , 76
lvarpush , 87	saveprocs , 82	tk_focusPrev , 77
max , 87	scale , 58	tk_getOpenFile , 77
menu , 47	scan , 17	tk_getSaveFile , 77
menubutton , 55	scancontext , 86	tk_menuSetFocus , 77
message , 56	scanfile , 86	tk_messageBox , 77
min , 87	scanmatch , 86	tk_optionMenu , 77
namespace , 25	scrollbar , 59	tk_popup , 77
nice , 83	seek , 22	tk_setPalette , 77
open , 22	select , 85	tk_textCopy , 78
option , 75	selection , 75	tk_textCut , 78
pack , 71	send , 76	tk_textPaste , 78
package , 24	set , 28	Tkspline , 44
package Img , 90	showproc , 80	Tktable , 98
package Tkspline , 44	signal , 83	tkwait , 76
package Tktable , 98	sleep , 83	toplevel , 62
package Vimconsole , 114	socket , 22	trace , 29
package vu , 105	source , 28	translit , 89
panedwindow , 56	spinbox , 60	try_eval , 81
parray , 14	split , 14	umask , 83
photo , 63	string , 17	
pid , 20	subst , 19	
	switch , 9	
	sync , 83	

union , 88	vim , 112	vwait , 30
unknown , 29	Vim if_tcl , 112	wait , 83
unset , 29	Vimconsole , 114	while , 9
update , 29	vu , 105	winfo , 65
uplevel , 29	vu::bargraph , 105	wish , 31
upvar , 30	vu::dial , 107	wm , 67
variable , 26	vu::pie , 109	write_file , 85

