Notes

Tcl/Tk Reference Guide

for Tcl 8.0 / Tk 8.0

Tk/Tcl program designed and created by
John Ousterhout **<ouster@scriptics.com>**

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

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

Contents

Rev. 8.0.3

Conventions

fixed	denotes literal text.
this	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. Basic Tcl Language Features

; or *newline*	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 (*d*=0-7)
n	newline (0xA)	\x*dd*	hexadecimal value (*d*=0-9, a-f)
r	carriage return (0xD)	*c*	replace '*c*' with '*c*'
t	horizontal tab (0x9)	\\	a backslash
v	vertical tab (0xB)		

2. Tcl Special Variables

argc	Number of command line arguments.
argv	Tcl list of command line arguments.
arg0	Name of script or command interpreter.
env	Array where each element name is an enviroment variable.
errorCode	Error code information from the last Tcl error.
errorInfo	Describes the stack trace of the last Tcl error.
tcl_library	Location of standard Tcl libraries.
tcl_patchLevel	Current patchlevel of Tcl interpreter.
tcl_pkgPath	List of directories to search for package loading.
tcl_platform	Array with elements **byteOrder**, **osVersion**, **machine**, **platform**, and **os**.

Command Index

-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	**selectForeground**
highlightBackground	**troughColor**

tcl_precision	Number of significant digits to retain when converting floating-point numbers to strings (default 12).
tcl_traceCompile	Level of tracing info output during bytecode compilation.
tcl_traceExec	Level of tracing info output during bytecode execution.
tcl_version	Current version of Tcl interpreter.

3. 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
&	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, 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 comands in brackets, and the following math functions:

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

4. Regular Expressions

regex|*regex*	match either expression
*regex**	match zero or more of *regex*
regex+	match one or more of *regex*
regex?	match zero or one of *regex*
.	any single character except newline
^	match beginning of string
$	match end of string
c	match character *c* even if special
[abc]	match set of characters
[^abc]	match characters not in set
[a-z]	match range of characters
[^a-z]	match characters not in range

( )	group expressions

5. Pattern Globbing

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

6. 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** *code*] [*string*]
Return immediately from current procedure with *string* as return value.

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 **-exact**, **-glob**, and **-regexp**.

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

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_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*.
-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_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.

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*.

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 scaling [**-displayof** *window*] [*floatNumber*]
Set scaling factor for conversion between physical units and pixels on *window*'s display where *floatNumber* is pixels per point (1/72 inch).

7. File Information

file atime *fileName*
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**, **-owner**, **-permissions**; for Windows **-archive**, **-hidden**, **-longname**, **-readonly**, **-shortname**, **-system**; and for MacOS: **-creator**, **-hidden**, **-readonly**, **-type**.

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

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

file mtime *fileName*
Time *fileName* was last modified as seconds since Jan. 1, 1970.

file nativename *fileName*
Returns the platform-specific name of *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 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 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 volume
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.

8. 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*. 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 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*.

32. 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. *option value* [*option value* ...]
A Tcl list of *option/value*s as valid for **font create**.

Font Options:

-family *name*	Font family (e.g. **Courier**, **Times**, **Helvetica**).
-size *size*	Size in points (or pixels if negative).
-weight *weight*	Either **normal** (default) or **bold**.
-slant *slant*	Either **roman** (default) or **italic**.
-underline *boolean*	Whether or not font is underlined.
-overstrike *boolean*	Whether or not font is overstriked.

33. Other Tk Commands

bell [**-displayof** *window*]
Ring the X bell on *window*'s display.

clipboard clear [**-displayof** *window*]
Claim ownership of clipboard on *window*'s display, clearing its contents.

clipboard append [**-displayof** *win*] [**-format** *fmt*] [**-type** *type*] *data*
Append *data* to clipboard on *win*'s display.

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* [*slave* ...] [*option value* ...]
Sets how slave windows should be managed by grid geometry master.

-column *n*	**-ipady** *amount*	**-row** *n*
-columnspan *n*	**-padx** *amount*	**-rowspan** *n*
-in *other*	**-pady** *amount*	**-sticky** [**n**][**s**][**e**][**w**]
-ipadx *amount*		

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.
-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.

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
Returns 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 *****).

9. Lists

concat [*arg arg* ...]
Returns concatenation of each list *arg* as a single list.

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

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 [*mode*] *list pattern*
Returns index of first element in *list* that matches *pattern* (-1 for no match). Mode may be **-exact**, **-glob** (default), or **-regexp**.

lsort [*switches*] *list*
Returns new list formed by sorting *list* according to *switches*. These are

-ascii	string comparison (default)
-dictionary	like **-ascii** but ignores case and is number smart.
-index *ndx*	treats each elements as a sub-list and sorts on the *ndx*th element
-integer	integer comparison

-real	floating-point comparison
-increasing	sort in increasing order (default)
-decreasing	sort in decreasing order
-command *cmd*	Use *cmd* which takes two arguments and returns an integer less than, equal to, or greater than zero

split *string* [*splitChars*]
Returns a list formed by splitting *string* at each character in *splitChars*.

Note: list indices start at 0 and the word **end** may be used to reference the last element in the list.

10. 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* [*pattern*]
Returns list of all element names in *arrayName* that match glob *pattern*.

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*.

parray *arrayName* [*pattern*]
Print to standard output the names and values of all element names in *arrayName* that match glob *pattern*.

11. 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 *arg*s composed according to *formatString*, a sequence of zero or more field codes each followed by an optional integer count. The possible field codes are:

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

-count *number*	%c	**Expose**
-detail *detail*	%d	**Enter, Leave, Focus**
-focus *boolean*	%f	**Enter, Leave**
-height *size*	%h	**Configure**
-keycode *number*	%k	**KeyPress, KeyRelease**
-keysym *name*	%K	**KeyPress, KeyRelease**
-mode *notify*	%m	**Enter, Leave, Focus**
-override *boolean*	%o	**Map, Reparent, Configure**
-place *where*	%p	**Circulate**
-root *window*	%R	†
-rootx *coord*	%X	†
-rooty *coord*	%Y	†
-sendevent *boolean*	%E	*all events*
-serial *number*	%#	*all events*
-state *state*	%s	*all events*
-subwindow *window*	%S	†
-time *integer*	%t	†, **Property**
-x *coord*	%x	†, §
-y *coord*	%y	†, §

† **KeyPress, KeyRelease, ButtonPress, ButtonRelease, Enter, Leave, Motion**
§ **Expose, Configure, Gravity, Reparent**

31. Geometry Management

The pack Command

pack [**configure**] *slave* [*slave* ...] [*options*]
Sets how slave windows should be managed by pack geometry master.

-after *sibling*	**-in** *master*	**-pady** *pixels*
-anchor *anchor*	**-ipadx** *pixels*	**-fill none**|**x**|**y**|**both**
-before *sibling*	**-ipady** *pixels*	**-side top**|**bottom**|**left**|**right**
-expand *boolean*	**-padx** *pixels*	

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 [**configure**] *window option value* [*option value* ...]
Sets how given windows should be placed inside their master.

-anchor *anchor*	**-relheight** *size*	**-rely** *location*
-height *size*	**-relwidth** *size*	**-x** *location*
-in *master*	**-relx** *location*	**-y** *location*
-width *size*	**-bordermode inside**|**outside**|**ignore**	

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 *sequence*s occur.

event delete **<<***virtual***>>** [*sequence* ...]
Deletes given *sequence*s (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:

Any	**Triple**	**Button5, B5**	**Mod3, M3**
Control	**Button1, B1**	**Meta, M**	**Mod4, M4**
Shift	**Button2, B2**	**Mod1, M1**	**Mod5, M5**
Lock	**Button3, B3**	**Mod2, M2**	**Alt**
Double	**Button4, B4**		

Types:

Activate	**Enter**	**Motion**
ButtonPress, Button	**Expose**	**Leave**
ButtonRelease	**FocusIn**	**Map**
Circulate	**FocusOut**	**Property**
Colormap	**Gravity**	**Reparent**
Configure	**KeyPress, Key**	**Unmap**
Deactivate	**KeyRelease**	**Visibility**
Destroy		

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
-above *window*	%a	**Configure**
-borderwidth *size*	%B	**Configure**
-button *number*	%b	**ButtonPress, ButtonRelease**

binary scan *string formatString varName* [*varName* ...]
Extracts values into *varName*'s from binary *string* according to *formatString*. Returns the number of values extracted. Field codes are the same as for **binary format** except for:

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

format *formatString* [*arg arg* ...]
Returns a formated string generated in the ANSI C **sprintf**-like manner. Placeholders have the form **%**[*argpos***$**][*flag*][*width*][**.***prec*][**h** | **l**]*char* where *argpos*, *width*, and *prec* are integers and 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		

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. The **-nocase** switch can be specified to ignore case in matching. The **-indices** switch can be specified so that *matchVar* and *subMatchVar* will be set to the start and ending indices in *string* of their corresponding match.

regsub [*switches*] *exp string subSpec varName*
Replaces the first portion of *string* that matches the regular expression *exp* with *subSpec* and places results in *varName*. Returns count of number of replacements made. The **-nocase** switch can be specified to ignore case in matching. The **-all** switch will cause all matches to be substituted for.

scan *string formatString varName* [*varName* ...]
Extracts values into given variables using ANSI C **sscanf** behavior. Returns the number of values extracted. Placeholders have the form **%**[*****][*width*]*char* where ***** is for discard, *width* is an integer and 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
c	char to int				

string compare *string1 string2*
Returns -1, 0, or 1, depending on whether *string1* is lexicographically less than, equal to, or greater than *string2*.

string first *string1 string2*
Return index in *string2* of first occurance of *string1* (-1 if not found).

string index *string charIndex*
Returns the *charIndex*'th character in *string*.

string last *string1 string2*
Return index in *string2* of last occurance of *string1* (-1 if not found).

string length *string*
Returns the number of characters in *string*.

string match *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.

string tolower *string*
Returns new string formed by converting all chars in *string* to lower case.

string toupper *string*
Returns new string formed by converting all chars in *string* to upper case.

string trim *string* [*chars*]
Returns new string formed by removing from *string* any leading or trailing characters present in the set *chars* (defaults to 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 [-nobackslashes] [-nocommands] [-novariables] *string*
Returns result of backslash, command, and variable substitutions on *string*. Each may be turned off by switch.

12. System Interaction

cd [*dirName*]
Change working directory to *dirName*.

clock clicks
Returns hi-res system-dependent integer time value.

clock format *clockVal* [**-format** *string*] [**-gmt** *boolean*]
Convert integer *clockVal* to human-readable format defined by *string* which recognizes (at least) the following placeholders:

%%	%	%H	hour (00 – 23)	%U	week (01 – 52)
%a	weekday (abbr)	%h	hour (00 – 12)	%w	weekday (0 – 6)
%A	weekday (full)	%j	day (001 – 366)	%x	locale date
%b	month (abbr)	%m	month (01 – 12)	%X	locale time
%B	month (full)	%M	minute (00 – 59)	%y	year (00 – 99)
%c	locale date & time	%p	AM/PM	%Y	year (full)
%d	day (01 – 31)	%S	seconds (00 – 59)	%Z	time zone

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.

wm group *window* [*pathName*]
Gives path name for leader of group to which *window* belongs.

wm iconbitmap *window* [*bitmap*]
Specifies a bitmap to use as icon image when *window* is iconified.

wm iconify *window*
Arrange for *window* to be iconified.

wm iconmask *window* [*bitmap*]
Specifies a bitmap to use to mask icon image when *window* is iconified.

wm iconname *window* [*newName*]
Specifies name to use as a label for *window*'s icon.

wm iconposition *window* [*x y*]
Specifies position on root window to place *window*'s icon.

wm iconwindow *window* [*pathName*]
Sets path name of window to use as the icon when *window* is iconified.

wm maxsize *window* [*width height*]
Specifies maximum size *window* may be resized to in each direction.

wm minsize *window* [*width height*]
Specifies minimum size *window* may be resized to in each direction.

wm overrideredirect *window* [*boolean*]
Set or unset the override-redirect flag of *window* commonly used by window manager to determine whether window should decorative frame.

wm positionfrom *window* [**program**|**user**]
Indicate from whom the *window*'s current position was requested.

wm protocol *window* [*name*] [*command*]
Specify a Tcl command to be invoked for messages of protocol *name*.

wm resizable *window* [*widthBoolean heightBoolean*]
Specifies whether *window*'s width and/or height is resizable.

wm sizefrom *window* [**program**|**user**]
Indicate from whom the *window*'s current size was requested.

wm state *window*
Returns current state of *window*: **normal**, **icon**, **iconic**, or **withdrawn**.

wm title *window* [*string*]
Set title for *window*'s decorative frame to *string*.

wm transient *window* [*master*]
Informs window manager that *window* is a transient of the window *master*.

wm withdraw *window*
Arranges for *window* to be withdrawn from the screen.

30. 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 script*
Binds *script* to the given sequence for *tag*. If *script* is the empty string, the binding is deleted. If the first character of *script* is a **+**, then it is appended to the currently associated script.

winfo screenwidth *window*
Returns the width in pixels of *window*'s screen.

winfo toplevel *window*
Returns the pathname of the top-level window containing *window*.

winfo visual *window*
Returns the visual class of *window* (see **winfo screenvisual**).

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.

29. The Window Manager

wm aspect *window [minNumer minDenom maxNumer maxDenom]*
Inform window manager of desired aspect ratio range for *window*.

wm client *window* [*name*]
Store *name* in *window*'s **WM_CLIENT_MACHINE** property. Informs window manager of client machine on which the application is running.

wm colormapwindows *window* [*windowList*]
Store *windowList* in *window*'s **WM_COLORMAP_WINDOWS** property which identifies the internal windows within *window* with private colormaps.

wm command *window* [*value*]
Store *value* in *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**]
Specifies 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*]
Changes geometry of *window* to *newGeometry*.

wm grid *window* [*baseWidth baseHeight widthInc heightInc*]
Indicates that *window* is to be managed as a gridded window with the specified relation between grid and pixel units.

exec [**-keepnew**] *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)
<*fileName*	stdin from file
<@ *fileId*	stdin from open file
<<*value*	pass value to stdin
>*fileName*	stdout to file
2**>***fileName*	stderr to file
>& *fileName*	stdout and stderr to file
>>*fileName*	append stdout to file
2**>>***fileName*	append stderr to file
>>& *fileName*	stdout and stderr to file
>@ *fileId*	stdout to open file
2**>@** *fileId*	stderr to open file
>&@ *fileId*	stdout and stderr to open file
&	run in background

glob [**-nocomplain**] *pattern* [*pattern ...*]
Returns list of all files in current directory that match any of the given csh-style glob patterns, optionally suppressing error on no match.

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

pwd Returns the current working directory.

13. 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.

-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):

-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, data bits, and stop bits of channel.

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* [*varName*]
Read next line from channel *fileId*, discarding newline character. Places characters of line in *varName* if given, otherwise returns them.

open *fileName* [*access*] [*perms*]
Opens *filename* and returns its channel id. If a new file is created, its permission are set to the conjuction of *perms* and the process umask. The *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.

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**, **current**, or **end**.

socket [*option ...*] *host port*
Open a client-side TCP socket to server *host* on *port*. 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* invoking *command* once connected with three arguments: the channel, the address, and the port number.

tell *fileId*
Return current access position in *fileId*.

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 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 pixels *window number*
Returns the number of pixels in *window* corresponding to the distance given by *number*, rounded to nearest integer.

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 server *window*
Returns server information on *window*'s display.

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**.

-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 **write** *fileName* [*option value* ...]
Writes image data from image into file *fileName*.

-format *format-name*
Specifies image format for the file.

-from *x1 y1 x2 y2*
Specifies a rectangular region of the image to copy from.

28. Window Information

winfo allmapped *window*
Returns 1 if *window* and all its ancestors are mapped, 0 otherwise.

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 *width*x*height*+*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.

14. 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 *arg*s 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* ...]
Evalutes concatenation of *arg*s as command in interpreter *path*.

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

interp expose *path hiddenCmd* [*exposedCmd*]
Make *hiddenCmd* in interp *path* exposed (optionally as *exposedCmd*).

interp hide *path exposedCmd* [*hiddenCmd*]
Make *exposedCmd* in interp *path* hidden (optionally as *hiddenCmd*).

interp hidden *path*
Returns list of hidden commands in interp *path*.

interp invokehidden *path* [**-global**] *hiddenCmd* [*arg* ...]
Invokes *hiddenCmd* with specified *arg*s in interp *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 interp *path* as trusted.

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**, and **marktrusted** 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**	**incr**	**lsearch**	**split**
append	**expr**	**info**	**lsort**	**string**
array	**fblocked**	**interp**	**package**	**subst**

break	**fileevent**	**join**	**pid**	**switch**
case	**flush**	**lappend**	**proc**	**tell**
catch	**for**	**lindex**	**puts**	**trace**
clock	**foreach**	**linsert**	**read**	**unset**
close	**format**	**list**	**rename**	**update**
concat	**gets**	**llength**	**return**	**uplevel**
continue	**global**	**lower**	**scan**	**upvar**
eof	**history**	**lrange**	**seek**	**vwait**
error	**if**	**lreplace**	**set**	**while**

A safe interpreter is created with the following commands hidden:

cd	**fconfigure**	**load**	**pwd**	**source**
exec	**file**	**open**	**socket**	**vwait**
exit	**glob**			

15. Packages

package forget *package*
Remove all info about *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.

16. 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.

-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*.

-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 **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.

imageName **get** *x y*
Returns RGB value of pixel at coords *x y* as list of three integers.

imageName **put** *data* [**-to** *x1 y1 x2 y2*]
Sets pixels values for the region *x1 y1 x2 y2* for 2-D array *data*.

imageName **read** *fileName* [*option value* ...]
Reads image data from file *fileName* 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.

Toplevel

-borderwidth	**-highlightbackground**	**-relief**
-cursor	**-highlightcolor**	**-takefocus**
-height	**-highlightthickness**	**-width**

-background *color*
Same as standard but my 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.

-use *windowID*
Toplevel should be embedded inside window identified by *windowID* (see **winfo id**) which was created as a container.

-screen *screen*
Screen on which to place the window.

-visual *visual*
Specifies visual to use for window.

27. 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 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.

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 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 *value*s. Inside a procedure and outsize a **namespace eval**, a local variable is created linked to the given namespace variable.

17. 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**.

bgerror *message*
User defined handler for background Tcl errors. Default exists for Tk.

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*.

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* from *file* (dynamic lib) 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 commmands to evaluate when invoked.

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

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

-command *tclCommandPrefix*
Prefix of a Tcl command to invoke 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.

-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 invoke the Tcl command specified with **-command**, if any.

radiobutton **select**
Select the radiobutton.

Scale

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

-bigincrement *number*
A real value to use for large increments of the scale.

-command *tclCommand*
Specified a TCL command to invoke 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.

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* and evaluate its contents as a Tcl *script*.

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

trace variable *varName ops command*
Arrange for *command* to be evaluated whenever *varName* is accessed in one of the ways specified with *ops*. Possbile values are **r** for read, **w** for written, **u** for unset, and any combination of the three.

trace vdelete *varName ops command*
Remove any previous trace specified with the given arguments.

trace vinfo *varName*
Returns list describing each trace on *varName*.

unknown *cmdName* [*arg arg* ...]
Called when the Tcl interpreter encounters an undefined command name.

unset *varName* [*varName* ...]
Removes the given variables and arrays from scope.

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.

18. 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.

-activeborderwidth *width*
Width in screen units of widget border when it is active.

-activeforeground *color*
Foreground color of widget when it is active.

-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 run 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).

-disabledforeground *color*
Foreground color of widget when it is disabled.

-exportselection *boolean*
Whether or not a selection in the widget should also be the X selection.

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**	

-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.

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.

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

-offvalue *value*
Value given to variable specified with **-variable** option when the checkbutton is deselected.

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

-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 any.

checkbutton **select**
Select the checkbutton.

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

Frame

-borderwidth	**-highlightbackground**	**-relief**
-cursor	**-highlightcolor**	**-takefocus**
-height	**-highlightthickness**	**-width**

-background *color*
Same as standard expect 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*
Colormap to use for the window if different from parent.

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

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

Label

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

-font *font*
Font to use when drawing text inside the widget.

-foreground *color*
Foreground color of widget in normal state (Abbrev: **-fg**).

-height *width* | *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.

-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 **normal** | **disabled** (| **active** for button-type widgets)
Current state of widget.

-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.

19. Tk Special Variables

tk_library
Directory containing library of standard Tk scripts.

tk_patchLevel
Integer specifying current patch level for Tk.

tkPriv
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_version
Current version of Tk in *major.minor* form.

20. Widget Scroll Commands

The Canvas, Listbox and Text widgets support the following scrolling commands. The Entry widget supports 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*
Shift the view by 10 times the difference between the coordinates *x* and *y* and the current view anchor coordinates.

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.

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

26. Other Standard Widgets

Button

-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**

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

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 any.

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.

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, -regexp, -count** *var*, **-nocase**

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* [*script*]]
Arrange for *script* 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*.

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) and Listbox Widget also supports the following:

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

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

21. 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.

Coordinate examples: 5 (pixel), 2.2i (inch), 4.1c (cm), 3m (mm), 21p (pts)

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

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

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

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* ...]
Query or modify the coordinates that define an item.

canvas **create** *type x y* [*x y* ...] [*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*.

Text Embedded Window Options

-align top|center|bottom|baseline
How window is vertically aligned with its line.

-create *script*
Script to create and return window pathname if no **-window** option is given.

-padx *width*
Extra space in screen units to leave on the left and right side of window.

-pady *height*
Extra space in screen units to leave on the top and bottom of window.

-stretch *boolean*
Whether window should be stretched vertically to fill line.

-window *pathName*
Name of window to display.

Text Embedded Image Options

-align top|center|bottom|baseline
Where image is displayed on the line.

-image *image*
Specifies Tk image to use for embedded image.

-name *imageName*
Specifies name which may be used to reference the embedded image.

-padx *width*
Extra space in screen units to leave on the left and right side of image.

-pady *height*
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 **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**, **-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 **get** *index1* [*index2*]
Returns string of characters in given range.

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*.

-variable *variable*
Name of global variable to set when checkbutton or radiobutton is selected.

25. 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**

-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**.

-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: ± *count* **chars**, ± *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 *bitmap* Stipple pattern for background.

-fgstipple *bitmap* Stipple pattern for foreground.

-lmargin1 *size* Left margin of first line of a paragraph.

-lmargin2 *size* Left margin of wrapped lines of a paragraph.

-offset *size* Offset of baseline from normal baseline.

-overstrike *boolean* Whether to overstrike text.

-rmargin *size* Right margin of all lines.

-tabs *tabList* Set of tab stops (see **-tabs** above).

-underline *boolean* Whether to underline text.

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*
Rescale 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*.

Canvas Item Types

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

-fill *color* **-stipple** *bitmap* **-width** *outlineWidth*
-outline *color* **-tags** *tagList*

-extent *degrees*
Size of the angular range occupied by arc.

-outlinestipple *bitmap*
Bitmap stipple to use to draw arc's outline.

-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* ...]

-anchor *anchorPos* **-bitmap** *bitmap* **-tags** *tagslist*
-background *color* **-foreground** *color*

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

-anchor *anchorPos* **-image** *image* **-tags** *tagslist*

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

-fill *color* **-stipple** *bitmap* **-width** *outlineWidth*
-smooth *boolean* **-tags** *tagList*

-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. Default is **butt**.

-joinstyle bevel|**miter**|**round**
How joints are to be drawn at vertices. Default is **miter**.

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

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

-fill *color* **-stipple** *bitmap* **-width** *outlineWidth*
-outline *color* **-tags** *tagList*

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

-fill *color* **-smooth** *boolean* **-tags** *tagList*
-outline *color* **-stipple** *bitmap* **-width** *outlineWidth*

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 submenu 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.

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

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

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

-indicatoron *boolean*
Whether indictor 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 checkbutton entry's associated variable when deselected.

-onvalue *value*
Value to store in checkbutton entry's associated variable when selected.

-selectcolor *color*
Color for indicator in checkbutton and radiobutton entries.

-selectimage *image*
Image to draw in indicator for checkbutton and radiobutton entries.

-value *value*
Value to store in radiobutton entry's associated variable when selected.

listbox **selection set** *first* [*last*]
Add all elements between *first* and *last* inclusive to selection.

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

listbox **size**
Returns number of elements in listbox.

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

24. The Menu Widget

Menu Widget Options

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

-postcommand *tclCommand*
Specify Tcl command to invoke immediately before the menu is posted.

-selectcolor *color*
Specifies indicator color for checkbutton 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, checkbutton, command, radiobutton, separator**

Menu Indices: *number*, **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*.

-splinesteps *number*
Degree of smoothness desired for curved perimeter.

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

-fill *color* **-outline** *color* **-stipple** *bitmap* **-tags** *tagList* **-width** *outlineWidth*

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

-anchor *anchorPos* **-fill** *color* **-font** *font* **-stipple** *bitmap* **-tags** *tagList* **-text** *string*

-justify **left** | **right** | **center**
How to justify text within its bounding region.

-width *lineLength*
Maximum line length for the text. If zero, break only on \n.

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

-anchor *anchorPos* **-tags** *tagList*

-height *height* Height in screen units to assign item's window.

-width *width* Width in screen units to assign item's window.

-window *pathName* Window to associate with item.

Canvas Postscript Options

-colormap *varName*
Specifies a color mapping to use where *varName* is an array variable whose elements specify Postscript code to set a particular color value.

-colormode **color** | **grey** | **mono**
Specifies how to output color information.

-file *fileName*
Specifies the name of the file in which to write the Postscript. If not specified, the Postscript is returned as the result of the command.

-fontmap *varName*
Specifies a font mapping to use where *varName* is an array variable whose elements specify the Postscript font and size to use as a two element list.

-height *size*
Specifies the height of the area of the canvas to print. Defaults to the height of the canvas window

-pageanchor *anchor*
Specifies which point of the printed area should be appear over the positioning point on the page. Defaults to **center**.

-pageheight *size*
Specifies that the Postscript should be scaled in both x and y so that the printed area is *size* high on the Postscript page.

-pagewidth *size*
Specifies that the Postscript should be scaled in both x and y so that the printed area is *size* wide on the Postscript page.

-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.

22. The Entry Widget

Entry Widget Options

-background	**-highlightcolor**	**-relief**
-borderwidth	**-highlightthickness**	**-selectbackground**
-cursor	**-insertbackground**	**-selectborderwidth**
-exportselection	**-insertborderwidth**	**-selectforeground**
-font	**-insertofftime**	**-state**
-foreground	**-insertontime**	**-takefocus**
-highlightbackground	**-insertwidth**	**-textvariable**
	-justify	**-width**

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

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 is 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.

23. The Listbox Widget

Listbox Widget Options

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

-selectMode **single**|**browse**|**multiple**|**extended**

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

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

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

listbox **selection anchor** *index*
Set the selection anchor to element at *index*.

listbox **selection clear** *first* [*last*]
Deselect elements between *first* and *last* inclusive.

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