Tcl Quick Reference Card

(for version 7.3)

Basic Language Features

; or *newline*	statement separator
#	comment
var	simple variable
var(*index*)	associative array variable
var(*i,j*)	multi-dimensional array variable
`$var`	variable substitution
`${var}xyz`	variable substitution
[expr 1+2]	command substitution
`\$`	backslash substitution (see below)
"hello `$a`"	quoting with substitution
{hello `$a`}	quoting no substitution (deferred evaluation)

Backslash Substitutions

\a	audible alert (0x7)
\b	backspace (0x8)
\f	form feed (0xC)
\n	newline (0xA)
\r	carriage return (0xD)
\t	horizontal tab (0x9)
\v	vertical tab (0xB)
space	space
newline	space
ddd	octal value (*d*=0-7)
\x*dd*	hexadecimal value (*d*=0-9, a-f)
c	replace '*c*' with '*c*'

Built-In Variables

argc	number of command line arguments
argv	command line arguments (list)
argv0	command name
auto_path	path for autoloaded commands
env	environment variables (array)
errorCode	information on last error (list)
errorInfo	stacktrace after error
tcl_interactive	set to 1 if running interactive Tcl shell
tcl_precision	number of significant floating point digits
tcl_prompt1	prompt string
tcl_prompt2	prompt string for incomplete commands

Library Procedures

auto_execok *cmd*
auto_load *cmd*
auto_mkindex *dir pattern pattern ...*
auto_reset
parray *arrayName*
unknown *cmd ?arg arg ...?*

Operators (in decreasing precedence)

- ~ !	unary minus, bitwise NOT, logical NOT
* / %	multiply, divide, remainder
+ -	add, substract
<< >>	logical left and arithmetic right shift
< > <= >=	boolean comparisons
== !=	boolean equal, not equal
&	bit-wise AND
^	bit-wise exclusive OR
|	bit-wise 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 on strings.

Math Functions

abs	absolute value
acos	arc cosine
asin	arc sine
atan	arc tangent
atan2	arc tangent of x/y
ceil	round up to nearest integer
cos	cosine
cosh	hyperbolic cosine
double	convert to floating point
exp	e^x
floor	round down to nearest integer
fmod	floating point remainder of x/y
hypot	$\sqrt{x^2+y^2}$
int	convert to integer by truncation
log	natural logarithm
log10	base 10 logarithm
pow	x^y
round	convert to integer by rounding
sin	sine
sinh	hyperbolic sine
sqrt	square root
tan	tangent
tanh	hyperbolic tangent

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 line
`$`	match end of line
c	match character *c*
c	match character *c*
[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

Filename 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
~*user*	match *user*'s home directory

A "." at the beginning of a file's name or just after "/" must be matched explicitly. All "/" characters must be matched explicitly.

Features of exec Command

|	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*	stdour and stderr to open file
&	run in background

Written by Jeff Tranter 29-Jun-1994

Built-In Commands

append *varName value* ?*value value* ... ?
array anymore *arrayName searchId*
array donesearch *arrayName searchId*
array names *arrayName*
array nextelement *arrayName searchId*
array size *arrayName*
array startsearch *arrayName*
break
case - obsolete, see switch
catch *script* ?*varName*?
cd ?*dirName*?
close *fileId*
concat ?*arg arg* ...?
continue
eof *fileId*
error *message* ?*info*? ?*code*?
eval *arg* ?*arg* ...?
exec ?*switches*? *arg* ?*arg* ...?
    ?*switches*? = -keepnewline or - -
exit ?*returnCode*?
expr *arg* ?*arg arg* ...?
file atime *name*
file dirname *name*
file executable *name*
file exists *name*
file extension *name*
file isdirectory *name*
file isfile *name*
file lstat *name varName*
file mtime *name*
file owned *name*
file readable *name*
file readlink *name*
file rootname *name*
file size *name*
file stat *name varName*
file tail *name*
file type *name*
file writable *name*
flush *fileId*
for *start test next body*
foreach *varName list body*
format *formatString* ?*arg arg* ...?
gets *fileId* ?*varName*?
glob ?*switches*? *pattern* ?*pattern* ...?
    ?*switches*? = -nocomplain or - -
global *varName* ?*varName* ...?
history
history add *command* ?exec?
history change *newValue* ?*event*?
history event ?*event*?
history info ?*count*?
history keep ?*count*?
history nextid
history redo *event*
history substitute *old new* ?*event*?
history words *selector* ?*event*
if *expr1* ?then? *body1* elseif *expr2* ?then? *body2* elseif ... ?else? ?*bodyN*?
incr *varName* ?*increment*?
info args *procName*
info body *procName*
info cmdcount
info commands ?*pattern*?
info complete *command*
info default *procname arg varname*
info exists *varName*
info globals ?*pattern*?
info level ?*number*?
info library
info locals ?*pattern*?
info patchlevel
info procs ?*pattern*?
info script
info tclversion
info vars ?*pattern*?
join *list* ?*joinString*?
lappend *varName value* ?*value value* ...?
lindex *list index*
linsert *list index element* ?*element element* ...?
list ?*arg arg* ...?
llength *list*
lrange *list first last*
lreplace *list first last* ?*element element* ...?
lsearch ?*mode*? *list pattern*
lsort ?*switches*? *list*
open *fileName* ?*access*? ?*permissions*?
pid ?*fileId*?
proc *name args body*
puts ?-nonewline? ?*fileId*? *string*
pwd
read ?-nonewline? *fileId*
read *fileId numBytes*
regexp ?*switches*? *exp string* ?matchVar? ?subMatchVar ...?
    ?*switches*? = -indices, -nocase or - -
regsub ?*switches*? *exp string subSpec varName*
    ?*switches*? = -nocase or - -
rename *oldName newName*
return ?-code *code*? ?*string*?
return -code error ?-errorinfo *info*? ?-errorcode *code*? ?*string*?
scan *string format varName* ?*varName* ...?
seek *fileId offset* ?*origin*?
set *varName* ?*value*?
source *fileName*
split *string* ?*splitChars*?
string compare *string1 string2*
string first *string1 string2*
string index *string charIndex*
string last *string1 string2*
string length *string*
string match *pattern string*
string range *string first last*
string tolower *string*
string toupper *string*
string trim *string* ?*chars*?
string trimleft *string* ?*chars*?
string trimright *string* ?*chars*?
switch ?*options*? *string pattern body* ?*pattern body* ...?
switch ?*options*? *string* {*pattern body* ?*pattern body* ...?}
    ?*options*? = -exact, -glob, -regexp or - -
tell *fileId*
time *script* ?*count*?
trace variable *name ops command*
trace vdelete *name ops command*
trace vinfo *name*
unknown *cmdName* ?*arg arg* ...?
unset *name* ?*name name* ...?
uplevel ?*level*? *arg* ?*arg* ...?
upvar ?*level*? *otherVar myVar* ?*otherVar myVar* ...?
while *test body*