NAME

array − Manipulate array variables

SYNOPSIS

array *option arrayName* ?*arg arg ...*?

DESCRIPTION

This command performs one of several operations on the variable given by *arrayName*. Unless otherwise specified for individual commands below, *arrayName* must be the name of an existing array variable. The *option* argument determines what action is carried out by the command. The legal *options* (which may be abbreviated) are:

array anymore *arrayName searchId*

Returns 1 if there are any more elements left to be processed in an array search, 0 if all elements have already been returned. *Sear chId* indicates which search on *arrayName* to check, and must have been the return value from a previous invocation of **array startsearch**. This option is particularly useful if an array has an element with an empty name, since the return value from **array nextelement** will not indicate whether the search has been completed.

array donesearch *arrayName searchId*

This command terminates an array search and destroys all the state associated with that search. *SearchId* indicates which search on *arrayName* to destroy, and must have been the return value from a previous invocation of **array startsearch**. Returns an empty string.

array exists *arrayName*

Returns 1 if *arrayName* is an array variable, 0 if there is no variable by that name or if it is a scalar variable.

array get *arrayName* ?*pattern*?

Returns a list containing pairs of elements. The first element in each pair is the name of an element in *arrayName* and the second element of each pair is the value of the array element. The order of the pairs is undefined. If *pattern* is not specified, then all of the elements of the array are included in the result. If *pattern* is specified, then only those elements whose names match *pattern* (using the matching rules of **string match**) are included. If *arrayName* is not the name of an array variable, or if the array contains no elements, then an empty list is returned. If traces on the array modify the list of elements, the elements returned are those that exist both before and after the call to **array get**.

array names *arrayName* ?*mode*? ?*pattern*?

Returns a list containing the names of all of the elements in the array that match *pattern*. *Mode* may be one of **−exact**, **−glob**, or **−regexp**. If specified,*mode* designates which matching rules to use to match *pattern* against the names of the elements in the array. If not specified,*mode* def aults to **−glob**. See the documentation for**string match** for information on glob style matching, and the documentation for **regexp** for information on regexp matching. If *pattern* is omitted then the command returns all of the element names in the array. If there are no (matching) elements in the array, or if *arrayName* is not the name of an array variable, then an empty string is returned.

array nextelement *arrayName searchId*

Returns the name of the next element in *arrayName*, or an empty string if all elements of *arrayName* have already been returned in this search. The *searchId* argument identifies the search, and must have been the return value of an **array startsearch** command. Warning: if elements are added to or deleted from the array, then all searches are automatically terminated just as if **array donesearch** had been invoked; this will cause **array nextelement** operations to fail for those searches.

array set *arrayName list*
Sets the values of one or more elements in *arrayName*. *list* must have a form like that returned by **array get**, consisting of an even number of elements. Each odd-numbered element in *list* is treated as an element name within *arrayName*, and the following element in *list* is used as a new value for that array element. If the variable *arrayName* does not already exist and *list* is empty, *arrayName* is created with an empty array value.

array size *arrayName*
Returns a decimal string giving the number of elements in the array. If*arr ayName* is not the name of an array then 0 is returned.

array startsearch *arrayName*
This command initializes an element-by-element search through the array given by *arrayName*, such that invocations of the **array nextelement** command will return the names of the individual elements in the array. When the search has been completed, the**array donesear ch** command should be invoked. The return value is a search identifier that must be used in **array nextelement** and **array donesearch** commands; it allows multiple searches to be underway simultaneously for the same array. It is currently more efficient and easier to use either the **array get** or **array names**, together with **foreach**, to iterate over all but very large arrays. See the examples below for how to do this.

array statistics *arrayName*
Returns statistics about the distribution of data within the hashtable that represents the array. This information includes the number of entries in the table, the number of buckets, and the utilization of the buckets.

array unset *arrayName* ?*pattern*?
Unsets all of the elements in the array that match *pattern* (using the matching rules of **string match**). If*arr ayName* is not the name of an array variable or there are no matching elements in the array, no error will be raised. If *pattern* is omitted and *arrayName* is an array variable, then the command unsets the entire array. The command always returns an empty string.

EXAMPLES

```
array set colorcount {
   red   1
   green 5
   blue  4
   white 9
}

foreach {color count} [array get colorcount] {
   puts "Color: $color Count: $count"
}
 → Color: blue Count: 4
   Color: white Count: 9
   Color: green Count: 5
   Color: red Count: 1

foreach color [array names colorcount] {
   puts "Color: $color Count: $colorcount($color)"
}
 → Color: blue Count: 4
   Color: white Count: 9
   Color: green Count: 5
   Color: red Count: 1
```

```
foreach color [lsort [array names colorcount]] {
    puts "Color: $color Count: $colorcount($color)"
}
  → Color: blue Count: 4
    Color: green Count: 5
    Color: red Count: 1
    Color: white Count: 9

array statistics colorcount
  → 4 entries in table, 4 buckets
    number of buckets with 0 entries: 1
    number of buckets with 1 entries: 2
    number of buckets with 2 entries: 1
    number of buckets with 3 entries: 0
    number of buckets with 4 entries: 0
    number of buckets with 5 entries: 0
    number of buckets with 6 entries: 0
    number of buckets with 7 entries: 0
    number of buckets with 8 entries: 0
    number of buckets with 9 entries: 0
    number of buckets with 10 or more entries: 0
    average search distance for entry: 1.2
```

SEE ALSO

list(n), string(n), variable(n), trace(n), foreach(n)

KEYWORDS

array, element names, search