NAME

TclBin, bin_new, bin_def, bin_sizeof, bin_info, bin_move, bin_resize, bin_read, bin_write, bin_copy – Binary object manipulation

SYNOPSIS

bin_new *o:objectname s:typename i:sizeExpr* ?*o:father*? ?*i:offsetExpr*? ?*-no_unset*?

bin_def *o:object s:field s:type i:offsetExpr*

bin_sizeof *type*|*object*

bin_info *o:object*

bin_move ?*-absolute*? ?*-noerr*? *o:object* ?*i:offsetExpr*? ?*i:sizeExpr*?

bin_resize *o:object i:newSizeExpr*

bin_read ?*-nobuf*? *fileId o:object*

bin_write ?*-nobuf*? *fileId o:object*

bin_copy ?*-swapb*|*-rev4*? *o:objectDest o:objectSrc*|*byteConst* ?*i:offsetDestExpr*? ?*i:offsetSrcExpr*? ?*i:sizeExpr*?

DESCRIPTION

These procedures provides access to binary objects and C structures from Tcl.

Viewed from Tcl each C object is a Tcl array, each defined fields is a sub object. Trace functions provide the read/write conversions between Tcl strings and the Binary representation.

See also libtclbin.ps/.dvi for an introduction and overall description.

bin_new

Create a new object named *objectname* of type *typename*. The object created is *sizeExpr* bytes large. If *father* is given, no new memory is allocated and the object is then just a pointer to a (sub)space of father data.

If the type is a registered type plus a star (*), like *int**, the object is considered like an array of the base type. And accessing *objectname*(**i**) where **i** is a number will access the data at position **i** (starting at 0). Else, the typename is only an information string, that can be used to check the data allocated destination usage if needed.

For all objects, besides additional fields defined using bin_def(n), the following fields are defined:

*_obj_* Read only type name (all other fields can be used either to get the object content or to modify it).

*_str_* String representation of the object. Every instance of the character ascii value zero ('\0') is replaced by "*0*" and the backslash itself (\) is replaced by "\\".

*_hex_* Hexadecimal representation of the object: a string of twice the size of the object as an hexadecimal number.

i,*hex* Peek/poke a byte as a two characters hexadecimal number at offset **i** within the object.

i Access **i***th* element (starting at zero) of the object when it has been defined as an array of a built in type (*typename* is "*float**" for instance).

Returns the object name.

Memory is allocated for the new object (unless "father" is given).

PARAMETERS:

objectname
: (object) object name or "#auto" for automatically generated name

typename
: (string) a type name (like *char**, *int**, *double**, *mytype*,...)

sizeExpr
: (integer) expression giving the size in bytes of the object

father
: (optional object) father object name (container)

offsetExpr
: (optional integer) expression giving the offset in bytes from father start, ' where to attach this object

-no_unset
: (optional) if this argument is present, the object will not be automatically unset when father is unset (dangerous)

bin_def

Define a field named within object. Field *field* is defined inside the binary object *object* (like C struct) and the corresponding TCL associative array field is attached.

TclBin provides the following built in types :

char
: character

int
: integer (machine dependant, same as C type)

short
: short integer (machine dependant, same as C type)

netint
: 32 bits unsigned integer number, in standard network order (high byte first)

netshort
: 16 bits unsigned integer number, in standard network order.

double
: double precision floating point number (machine dependant, same as C type (in general IEEE)).

float
: simple precision floating point number (same as C).

str
: Null terminated, variable length (but not going outside object), array of characters.

It returns the string "*object(field)*".

PARAMETERS:

object
: (object) base object name (array name)

field
: (string) field name

type
: (string) registered type name (built in)

offsetExpr
: (integer) offset in bytes from base object first byte

bin_sizeof

returns the size in bytes of a type or object.

PARAMETERS:

type|object
name of object or type which size is wanted

bin_info
Dumps the Bin_Object structure of *object* as a keyed list. The following keys with values are returned:

data Memory data pointer address (should not be used beside debugging)

size Size in bytes (given more easily by bin_sizeof(n))

father Father object pointer address (or NULL)

type Type name. Also given by the *_obj_* field (which is not available for fields, so this is currently the only normal use of this command, beside debugging: to get the object type).

elemsize
Size of one element (in case of arrays)

deleteproc, peekpokeproc, traceproc
Internal callbacks used in object management (see "C API" section)

PARAMETERS:

object (object) name of object

bin_move
Move (incr by 1 object position, by default) a pointer object or a field. The new position, in bytes, is : *pos+sizeof(object)* by default, *pos+offsetExpr*sizeof(object)* when *offsetExpr* is given but *sizeExpr* is not given, or *pos+offsetExpr*sizeExpr* if both *offsetExpr* and *sizeExpr* are given. *pos* being the current position or father start when −*absolute* flag is used.

Returns the resulting absolute offset within father.

PARAMETERS:

-absolute
(optional) If the −*absolute* flag is used, the offset expression is absolute

-noerr (optional) If the −*noerr* flag is used, trying to move the object outside father limits will not generate an error.

object (object) object name

offsetExpr
(optional integer) offset expression

sizeExpr
(optional integer) size expression

bin_resize
Changes the size field of an object to *newSizeExpr*. Object must be a pointer object.

Returns the new size.

PARAMETERS:

object (object) object name

newSizeExpr
(integer) size expression

bin_read
Reads "*object'*"s size bytes from *fileId* into the object.

Returns the number of bytes actually read (can be less than object's size if channel configured non blocking and writing more would block for instance).

PARAMETERS:

-nobuf (optional) If present will use unbuffered (read(2)) call instead of stdio (fread(3)) call. For Tcl versions 7.5 or above this flags does not exist as only Tcl_Read(3) can be used.

fileId Any tcl file descriptor (Channel) open for reading.

object (object) object name

bin_write

Writes "*object'*"s size bytes from *object* to *fileId*.

Returns the number of bytes actually written.

PARAMETERS:

-nobuf (optional) If present will use unbuffered (write(2)) call instead of stdio (fwrite(3)) call. For Tcl versions 7.5 or above Tcl_Write(3) is always used but Tcl_Flush(3) is called if this flags is present.

fileId Any tcl file descriptor (Channel) open for writing.

object (object) object name

bin_copy

Copy (like memmove(3) (so areas can overlap)) memory from an object (source) to another one (destination). Checks source and destination size and copy maximum with these constraints, or the size specified by *sizeExpr*. If *offsetDestExpr* or *offsetSrcExpr* the copy starts with the given offset or else from objects.

If source is not an object but rather an integer number, this byte value is then copied over the destination object (memset(3) like).

Returns the number of bytes actually copied.

PARAMETERS:

-swapb|*-rev4*
(optional) swap bytes if −*swapb* is given, reverse 4 bytes order (1 2 3 4 to 4 3 2 1) if −*rev4*

objectDest
(object) destination

objectSrc|*byteConst*
(object) source object or byte constant

offsetDestExpr
(optional integer) offset within destination

offsetSrcExpr
(optional integer) offset within source (ignored for constant)

sizeExpr
(optional integer) number of bytes copied

C API

From C, each object is a Bin_Object structure with the following fields:

magic (int)
Magic number used to check integrity of the objects when the library has been compiled with VERIFYSTRUCT.

*data (void *)*
Pointer to the actual data of the object

size (size_t)
data size (used for read/write & control)

*type (char *)*
pointer to the type name string (allocated at the end of the structure)

*father (Bin_Object *)*
father object (if object is a sub object)

elemsize (size_t)
size of a single element (if applicable : if type is an array of built in types)

*DeleteProc (tfunc_obj *)*
function to be called when deleting the object (ckfree)

*PeekPokeProc (Tcl_VarTraceProc *)*
tcl trace proc to be called for reading/writing one element of that type (if applicable)

*TraceProc (Tcl_VarTraceProc *)*
tcl trace proc to be called for reading/writing that type (Bin_TraceObject(3) for plain objects)

See individual man pages for Bin_NewObject(3), Bin_Def(3), ... for specific API informations.

EXAMPLES

bin_new example

```
bin_new tab double* 10*[bin_sizeof double]
bin_new el2 double 8 tab1 2*8
puts $tab(_obj_) ;
puts [expr 0x$tab(3,hex)+3] ;
set tab(3,hex) 2;
puts $tab(0);
```

bin_def example

```
bin_new toto double* [bin_sizeof double]*10
bin_def toto dbl double 0
set toto(dbl) 5.6
```

bin_move example

```
bin_move el2
bin_move -absolute el2 2 1
```

bin_resize example

```
bin_new buf  buffer 128
bin_new line char*  16 buffer;
bin_resize line 32
```

BUGS

none known.

AUTHOR

Laurent Demailly, <ld@mail.box.eu.org> (pobox <dl@mail.dotcom.fr>) / <URL:http://www.box.eu.org/~dl/>

VERSION

Version 2.13 (1997/01/27)

CALLS

FILE

tclbin/bindata.c