NAME

xdr_admin, xdr_control, xdr_getpos, xdr_inline, xdrrec_endofrecord, xdrrec_eof, xdrrec_readbytes, xdrrec_skiprecord, xdr_setpos, xdr_sizeof – library routines for external data representation

MT-LEVEL

Safe

DESCRIPTION

XDR library routines allow C programmers to describe arbitrary data structures in a machine-independent fashion. Protocols such as remote procedure calls (RPC) use these routines to describe the format of the data.

These routines deal specifically with the management of the XDR stream.

Routines

See **rpc**(3N) for the definition of the **XDR** data structure. Note that any buffers passed to the XDR routines must be properly aligned. It is suggested that **malloc**(3C) be used to allocate these buffers or that the programmer insure that the buffer address is divisible evenly by four.

#include <rpc/xdr.h>

bool_t xdr_control(XDR ∗*xdrs*, **int** *req*, **void** ∗*info*);

A function macro to change or retrieve various information about an XDR stream. *req* indicates the type of operation and *info* is a pointer to the information. The supported values of *req*, their argument types and what they do are:

XDR_GET_BYTES_AVAIL	**xdr_bytesrec ∗**	return number of bytes left unconsumed in the stream and a flag indicating whether or not this is the last fragment.

u_int xdr_getpos(const XDR ∗*xdrs*);

A macro that invokes the get-position routine associated with the XDR stream, *xdrs*. The routine returns an unsigned integer, which indicates the position of the XDR byte stream. A desirable feature of XDR streams is that simple arithmetic works with this number, although the XDR stream instances need not guarantee this. Therefore, applications written for portability should not depend on this feature.

long ∗**xdr_inline(XDR** ∗*xdrs*, **const int** *len*);

A macro that invokes the in-line routine associated with the XDR stream, *xdrs*. The routine returns a pointer to a contiguous piece of the stream's buffer; *len* is the byte length of the desired buffer. Note: pointer is cast to **long ∗**.

Warning: **xdr_inline( )** may return **NULL** (**0**) if it cannot allocate a contiguous piece of a buffer. Therefore the behavior may vary among stream instances;

it exists for the sake of efficiency, and applications written for portability should not depend on this feature.

bool_t xdrrec_endofrecord(XDR * xdrs , int *sendnow***);**

This routine can be invoked only on streams created by **xdrrec_create( )** (see **xdr_create**(3N)). The data in the output buffer is marked as a completed record, and the output buffer is optionally written out if *sendnow* is non-zero. This routine returns TRUE if it succeeds, FALSE otherwise.

bool_t xdrrec_eof(XDR **xdrs***);**

This routine can be invoked only on streams created by **xdrrec_create( )**. After consuming the rest of the current record in the stream, this routine returns **TRUE** if there is no more data in the stream's input buffer. It returns **FALSE** if there is additional data in the stream's input buffer.

int xdrrec_readbytes(XDR **xdrs***, caddr_t** *addr***, u_int** *nbytes***);**

This routine can be invoked only on streams created by **xdrrec_create**( ). It attempts to read *nbytes* bytes from the XDR stream into the buffer pointed to by *addr***.** On success this routine returns the number of bytes read, –1 on failure. A return value of 0 indicates an end of record.

bool_t xdrrec_skiprecord(XDR **xdrs***);**

This routine can be invoked only on streams created by **xdrrec_create( )** (see **xdr_create**(3N)). It tells the XDR implementation that the rest of the current record in the stream's input buffer should be discarded. This routine returns TRUE if it succeeds, FALSE otherwise.

bool_t xdr_setpos(XDR **xdrs***, const u_int** *pos***);**

A macro that invokes the set position routine associated with the XDR stream *xdrs*. The parameter *pos* is a position value obtained from **xdr_getpos( )**. This routine returns **TRUE** if the XDR stream was repositioned, and **FALSE** otherwise.

Warning: it is difficult to reposition some types of XDR streams, so this routine may fail with one type of stream and succeed with another. Therefore, applications written for portability should not depend on this feature.

unsigned long xdr_sizeof(xdrproc_t *func***, void** **data***);**

This routine returns the number of bytes required to encode *data* using the XDR filter function *func*, excluding potential overhead such as RPC headers or record markers. **O** is returned on error. This information might be used to select between transport protocols, or to determine the buffer size for various lower levels of RPC client and server creation routines, or to allocate storage when XDR is used outside of the RPC subsystem.

SEE ALSO **malloc**(3C), **rpc**(3N), **xdr_complex**(3N), **xdr_create**(3N), **xdr_simple**(3N)