NAME

xdr_simple, xdr_bool, xdr_char, xdr_double, xdr_enum, xdr_float, xdr_free, xdr_hyper, xdr_int, xdr_long, xdr_longlong_t, xdr_quadruple, xdr_short, xdr_u_char, xdr_u_hyper, xdr_u_int, xdr_u_long, xdr_u_longlong_t, xdr_u_short, xdr_void – library routines for external data representation

MT-LEVEL

Safe

DESCRIPTION

XDR library routines allow C programmers to describe simple 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 require the creation of XDR streams (see **xdr_create**(3N)).

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_bool(XDR **xdrs*, **bool_t** **bp***);**

> **xdr_bool( )** translates between booleans (C integers) and their external representations. When encoding data, this filter produces values of either **1** or **0**. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_char(XDR **xdrs*, **char** **cp***);**

> **xdr_char( )** translates between C characters and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise. Note: encoded characters are not packed, and occupy 4 bytes each. For arrays of characters, it is worthwhile to consider **xdr_bytes( )**, **xdr_opaque( )**, or **xdr_string( )** (see **xdr_complex**(3N)).

bool_t xdr_double(XDR **xdrs*, **double** **dp***);**

> **xdr_double( )** translates between C **double** precision numbers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_enum(XDR **xdrs*, **enum_t** **ep***);**

> **xdr_enum( )** translates between C **enum**s (actually integers) and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_float(XDR **xdrs*, **float** **fp***);**

> **xdr_float( )** translates between C **float**s and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

void xdr_free(xdrproc_t *proc*, **char** **objp***);**

Generic freeing routine. The first argument is the XDR routine for the object being freed. The second argument is a pointer to the object itself. Note: the pointer passed to this routine is not freed, but what it points to is freed (recursively, depending on the XDR routine).

bool_t xdr_hyper(XDR **xdrs*, **longlong_t** **llp***);**

xdr_hyper( ) translates between ANSI C **long long** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_int(XDR **xdrs*, **int** **ip***);**

xdr_int( ) translates between C integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_long(XDR **xdrs*, **long** **lp***);**

xdr_long( ) translates between C **long** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_longlong_t(XDR **xdrs*, **longlong_t** **llp***);**

xdr_longlong_t( ) translates between ANSI C **long long** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise. This routine is identical to **xdr_hyper( )**.

bool_t xdr_quadruple(XDR **xdrs*, **long double** **pq***);**

xdr_quadruple( ) translates between IEEE quadruple precision floating point numbers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_short(XDR **xdrs*, **short** **sp***);**

xdr_short( ) translates between C **short** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_u_char(XDR **xdrs*, **unsigned char** **ucp***);**

xdr_u_char( ) translates between **unsigned** C characters and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_u_hyper(XDR **xdrs*, **u_longlong_t** **ullp***);**

xdr_u_hyper( ) translates between unsigned ANSI C **long long** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_u_int(XDR **xdrs*, **unsigned** **up***);**

A filter primitive that translates between a C **unsigned** integer and its external representation. This routine returns TRUE if it succeeds, FALSE otherwise.

bool_t xdr_u_long(XDR **xdrs*, **unsigned long** **ulp***);**

xdr_u_long( ) translates between C **unsigned long** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_u_longlong_t(XDR **xdrs*, **u_longlong_t** **ullp***);**

xdr_u_longlong_t( ) translates between unsigned ANSI C **long long** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise. This routine is identical to **xdr_u_hyper( )**.

bool_t xdr_u_short(XDR **xdrs*, **unsigned short** **usp***);**

xdr_u_short( ) translates between C **unsigned short** integers and their external representations. This routine returns **TRUE** if it succeeds, **FALSE** otherwise.

bool_t xdr_void(void);

This routine always returns **TRUE**. It may be passed to RPC routines that require a function parameter, where nothing is to be done.

SEE ALSO

malloc(3C), **rpc**(3N), **xdr_admin**(3N), **xdr_complex**(3N), **xdr_create**(3N)