NAME

ldns_buffer, ldns_buffer_new, ldns_buffer_new_frm_data, ldns_buffer_clear, ldns_buffer_printf, ldns_buffer_free, ldns_buffer_copy, ldns_buffer_export, ldns_buffer_export2str, ldns_buffer2str - buffers

SYNOPSIS

```
#include <stdint.h>
#include <stdbool.h>

#include <ldns/ldns.h>

ldns_buffer* ldns_buffer_new(size_t capacity);

void ldns_buffer_new_frm_data(ldns_buffer *buffer, const void *data, size_t size);

void ldns_buffer_clear(ldns_buffer *buffer);

int ldns_buffer_printf(ldns_buffer *buffer, const char *format, ...);

void ldns_buffer_free(ldns_buffer *buffer);

void ldns_buffer_copy(ldns_buffer* result, const ldns_buffer* from);

void* ldns_buffer_export(ldns_buffer *buffer);

char* ldns_buffer_export2str(ldns_buffer *buffer);

char* ldns_buffer2str(ldns_buffer *buffer);
```

DESCRIPTION

ldns_buffer

implementation of buffers to ease operations

ldns_buffers can contain arbitrary information, per octet. You can write to the current end of a buffer, read from the current position, and access any data within it.

Example use of buffers is in the source code of \ref host2str.c
struct ldns_struct_buffer

```
{
        The current position used for reading/writing:
        size_t   _position;

        The read/write limit:
        size_t   _limit;

        The amount of data the buffer can contain:
        size_t   _capacity;

        The data contained in the buffer:
        uint8_t *_data;

        If the buffer is fixed it cannot be resized:
        unsigned _fixed : 1;

        /** The current state of the buffer. If writing to the buffer fails
         * for any reason, this value is changed. This way, you can perform
         * multiple writes in sequence and check for success afterwards. */
        ldns_status _status;
};
typedef struct ldns_struct_buffer ldns_buffer;
```

ldns_buffer_new() creates a new buffer with the specified capacity.

capacity: the size (in bytes) to allocate for the buffer
Returns the created buffer

ldns_buffer_new_frm_data() creates a buffer with the specified data. The data IS copied and MEMORY allocations are done. The buffer is not fixed and can be resized using buffer_reserve().

buffer: pointer to the buffer to put the data in
data: the data to encapsulate in the buffer
size: the size of the data

ldns_buffer_clear() clears the buffer and make it ready for writing. The buffer's limit is set to the capacity and the position is set to 0.
buffer: the buffer to clear

ldns_buffer_printf() prints to the buffer, increasing the capacity if required using buffer_reserve(). The

buffer's position is set to the terminating '\\0' Returns the number of characters written (not including the terminating '\\0') or -1 on failure.

ldns_buffer_free() frees the buffer.
***buffer**: the buffer to be freed
Returns void

ldns_buffer_copy() Copy contents of the from buffer to the result buffer and then flips the result buffer. Data will be silently truncated if the result buffer is too small.
***result**: resulting buffer which is copied to.
***from**: what to copy to result.

ldns_buffer_export() Makes the buffer fixed and returns a pointer to the data. The caller is responsible for free'ing the result.
***buffer**: the buffer to be exported
Returns void

ldns_buffer_export2str() Exports and returns the data in the buffer as a null terminated char * string. The returned string must be freed by the caller. The buffer must be in write modus and may thus not have been flipped. The buffer is fixed after this function returns.

buffer: buffer containing char * data
Returns null terminated char * data, or NULL on error

ldns_buffer2str() Returns a copy of the data in the buffer as a null terminated char * string. The returned string must be freed by the caller. The buffer must be in write modus and may thus not have been flipped.

buffer: buffer containing char * data
Returns null terminated char * data, or NULL on error

AUTHOR

The ldns team at NLnet Labs.

REPORTING BUGS

Please report bugs to dns-team@nlnetlabs.nl or on GitHub at https://github.com/NLnetLabs/ldns/issues

COPYRIGHT

SEE ALSO

ldns_buffer_flip, *ldns_buffer_rewind*, *ldns_buffer_position*, *ldns_buffer_set_position*, *ldns_buffer_skip*, *ldns_buffer_limit*, *ldns_buffer_set_limit*, *ldns_buffer_capacity*, *ldns_buffer_set_capacity*, *ldns_buffer_reserve*, *ldns_buffer_at*, *ldns_buffer_begin*, *ldns_buffer_end*, *ldns_buffer_current*, *ldns_buffer_remaining_at*, *ldns_buffer_remaining*, *ldns_buffer_available_at*, *ldns_buffer_available*, *ldns_buffer_status*, *ldns_buffer_status_ok*, *ldns_buffer_write_at*, *ldns_buffer_write*, *ldns_buffer_write_string_at*, *ldns_buffer_write_string*, *ldns_buffer_write_u8_at*, *ldns_buffer_write_u8*, *ldns_buffer_write_u16_at*, *ldns_buffer_write_u16*, *ldns_buffer_read_at*, *ldns_buffer_read*, *ldns_buffer_read_u8_at*, *ldns_buffer_read_u8*, *ldns_buffer_read_u16_at*, *ldns_buffer_read_u16*, *ldns_buffer_read_u32_at*, *ldns_buffer_read_u32*, *ldns_buffer_write_u32*, *ldns_buffer_write_u32_at*. And **perldoc Net::DNS**, **RFC1034**, **RFC1035**, **RFC4033**, **RFC4034** and **RFC4035**.

REMARKS

This manpage was automatically generated from the ldns source code.