

**NAME**

**mdchain, md\_initm, md\_done, md\_append\_record, md\_next\_record, md\_get\_uint8, md\_get\_uint16, md\_get\_uint16be, md\_get\_uint16le, md\_get\_uint32, md\_get\_uint32be, md\_get\_uint32le, md\_get\_int64, md\_get\_int64be, md\_get\_int64le, md\_get\_mem, md\_get\_mbuf, md\_get\_uio** - set of functions to dissect an mbuf chain to various data types

**SYNOPSIS**

**options LIBMCHAIN** kldload libmchain

```
#include <sys/param.h>
```

```
#include <sys/uio.h>
```

```
#include <sys/mchain.h>
```

*void*

```
md_initm(struct mdchain *mdp, struct mbuf *m);
```

*void*

```
md_done(struct mdchain *mdp);
```

*void*

```
md_append_record(struct mdchain *mdp, struct mbuf *top);
```

*int*

```
md_next_record(struct mdchain *mdp);
```

*int*

```
md_get_uint8(struct mdchain *mdp, uint8_t *x);
```

*int*

```
md_get_uint16(struct mdchain *mdp, uint16_t *x);
```

*int*

```
md_get_uint16be(struct mdchain *mdp, uint16_t *x);
```

*int*

```
md_get_uint16le(struct mdchain *mdp, uint16_t *x);
```

*int*

```
md_get_uint32(struct mdchain *mdp, uint32_t *x);
```

*int*  
**md\_get\_uint32be**(*struct mdchain \*mdp, uint32\_t \*x*);

*int*  
**md\_get\_uint32le**(*struct mdchain \*mdp, uint32\_t \*x*);

*int*  
**md\_get\_int64**(*struct mdchain \*mdp, int64\_t \*x*);

*int*  
**md\_get\_int64be**(*struct mdchain \*mdp, int64\_t \*x*);

*int*  
**md\_get\_int64le**(*struct mdchain \*mdp, int64\_t \*x*);

*int*  
**md\_get\_mem**(*struct mdchain \*mdp, caddr\_t target, int size, int type*);

*int*  
**md\_get\_mbuf**(*struct mdchain \*mdp, int size, struct mbuf \*\*m*);

*int*  
**md\_get\_uio**(*struct mdchain \*mdp, struct uio \*uiop, int size*);

## DESCRIPTION

These functions are used to decompose mbuf chains to various data types. The *mdchain* structure is used as a working context and should be initialized through a call of the **mb\_initm**() function. It has the following fields:

*md\_top* (*struct mbuf \**) A pointer to the top of the parsed mbuf chain.

*md\_cur* (*struct mbuf \**) A pointer to the currently parsed mbuf.

*md\_pas*  
 (*int*) Offset in the current mbuf.

The **md\_done**() function disposes of an mbuf chain pointed to by the *mdp->md\_top* field and sets the field to NULL.

The **md\_append\_record**() appends a new mbuf chain using *m\_nextpkt* field to form a single linked list of

mbuf chains. If the *mdp->md\_top* field is NULL, then this function behaves exactly as the **md\_initm()** function.

The **md\_next\_record()** function extracts the next mbuf chain and disposes the current one, if any. For a new mbuf chain it calls the **md\_initm()** function. If there is no data left the function returns ENOENT.

All **md\_get\_\***() functions perform an actual copy of the data from an mbuf chain. Functions which have **le** or **be** suffixes will perform conversion to the little- or big-endian data formats.

**md\_get\_mem()** function copies *size* bytes of data specified by the *source* argument from an mbuf chain. The *type* argument specifies the method used to perform a copy, and can be one of the following:

**MB\_MSYSTEM**

Use the **bcopy()** function.

**MB\_MUSER** Use the **copyin(9)** function.

**MB\_MINLINE** Use an "inline" loop which does not call any function.

If *target* is NULL, an actual copy is not performed and the function just skips the given number of bytes.

## RETURN VALUES

All *int* functions return zero if successful, otherwise an error code is returned.

*Note:* after failure of any function, an mbuf chain is left in the broken state and only the **md\_done()** function can safely be called to destroy it.

## EXAMPLES

```
struct mdchain *mdp;
struct mbuf *m;
uint16_t length;
uint8_t byte;

receive(so, &m);
md_initm(mdp, m);
if (md_get_uint8(mdp, &byte) != 0 ||
    md_get_uint16le(mdp, &length) != 0)
    error = EBADRPC;
mb_done(mdp);
```

**SEE ALSO**

mbchain(9), mbuf(9)

**AUTHORS**

This manual page was written by Boris Popov <*bp@FreeBSD.org*>.