NAME

mbuf, **m_get**, **m_getclr**, **m_gethdr**, **m_devget**, **m_copym**, **m_copypacket**, **m_copydata**, **m_copyback**, **m_cat**, **m_prepend**, **m_pullup**, **m_split**, **m_adj**, **m_free**, **m_freem**, **mtod**, **mtocl**, **cltom**, **MGET**, **MGETHDR**, **MEXTMALLOC**, **MEXTADD**, **MCLGET**, **M_COPY_PKTHDR**, **M_ALIGN**, **MH_ALIGN**, **M_LEADINGSPACE**, **M_TRAILINGSPACE**, **M_PREPEND**, **MCHTYPE**, **MEXTREMOVE**, **MFREE**, − functions and macros for managing memory used by networking code

SYNOPSIS

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

struct mbuf *
m_get(int nowait, int type);

struct mbuf *
m_getclr(int nowait, int type);

struct mbuf *
m_gethdr(int nowait, int type);

struct mbuf *
m_devget(char *buf, int totlen, int off0, struct ifnet *ifp,
        void (*copy) __P((const void *, void *, size_t)));

struct mbuf *
m_copym(struct mbuf *m, int off0, int len, int wait);

struct mbuf *
m_copypacket(struct mbuf *m, int how);

void
m_copydata(struct mbuf *m, int off, int len, caddr_t cp);

void
m_copyback(struct mbuf *m0, int off, int len, caddr_t cp);

void
m_cat(struct mbuf *m, struct mbuf *n);

struct mbuf *
m_prepend(struct mbuf *m, int len, int how);

struct mbuf *
m_pullup(struct mbuf *n, int len);

struct mbuf *
m_split(struct mbuf *m0, int len0, int wait);

void
m_adj(struct mbuf *mp, int req_len);

struct mbuf *
m_free(struct mbuf *m);

void
m_freem(struct mbuf *m);

int
mtod(struct mbuf *m, datatype);
```

u_long
mtocl(*void *datapointer*);

caddr_t
cltom(*u_long clusternum*);

void
MGET(*struct mbuf *m*, *int how*, *int type*);

void
MGETHDR(*struct mbuf *m*, *int how*, *int type*);

void
MEXTMALLOC(*struct mbuf *m*, *int len*, *int how*);

void
MEXTADD(*struct mbuf *m*, *caddr_t buf*, *int type*,
*void (*free) __P((caddr_t, u_int, void *))*, *void *arg*);

void
MCLGET(*struct mbuf *m*, *int how*);

void
M_COPY_PKTHDR(*struct mbuf *to*, *struct mbuf *from*);

void
M_ALIGN(*struct mbuf *m*, *int len*);

void
MH_ALIGN(*struct mbuf *m*, *int len*);

int
M_LEADINGSPACE(*struct mbuf *m*);

int
M_TRAILINGSPACE(*struct mbuf *m*);

void
M_PREPEND(*struct mbuf *m*, *int plen*, *int how*);

void
MCHTYPE(*struct mbuf *m*, *int type*);

void
MEXTREMOVE(*struct mbuf *m*);

void
MFREE(*struct mbuf *m*, *struct mbuf *n*);

DESCRIPTION

The **mbuf** functions and macros provide an easy and consistent way to handle a networking stack's memory management needs. An **mbuf** consists of a header and a data area. It is of a fixed size, `MSIZE` (defined in `<machine/param.h>`), which includes overhead. The header contains a pointer to the next **mbuf** in the **mbuf chain**, a pointer to the next **mbuf chain**, a pointer to the data area, the amount of data in this mbuf, its type and a `flags` field.

The `type` variable can signify:

MT_FREE	the mbuf should be on the ‘‘free’’ list
MT_DATA	data was dynamically allocated
MT_HEADER	data is a packet header
MT_SONAME	data is a socket name
MT_SOOPTS	data is socket options
MT_FTABLE	data is the fragment reassembly header
MT_CONTROL	mbuf contains ancillary (protocol control) data
MT_OOBDATA	mbuf contains out-of-band data.

The flags variable contains information describing the **mbuf**, notably:

M_EXT	has external storage
M_PKTHDR	is start of record
M_EOR	is end of record
M_CLUSTER	external storage is a cluster.

If an **mbuf** designates the start of a record (M_PKTHDR), its flags field may contain additional information describing the content of the record:

M_BCAST	sent/received as link-level broadcast
M_MCAST	sent/received as link-level multicast
M_LINK0, M_LINK1, MLINK2	three link-level specific flags.

An **mbuf** may add a single **mbuf cluster** of MCLBYTES bytes (also defined in <machine/param.h>), which has no additional overhead and is used instead of the internal data area; this is done when at least MINCLSIZE bytes of data must be stored.

m_get(*int nowait*, *int type*)
Allocates an mbuf and initializes it to contain internal data. The *nowait* parameter is a choice of M_WAIT / M_DONTWAIT from caller. M_WAIT means the call cannot fail, but may take forever. The *type* parameter is an mbuf type.

m_getclr(*int nowait*, *int type*)
Allocates an mbuf and initializes it to contain internal data, then zeros the data area. The *nowait* parameter is a choice of M_WAIT / M_DONTWAIT from caller. The *type* parameter is an mbuf type.

m_gethdr(*int nowait*, *int type*)
Allocates an mbuf and initializes it to contain a packet header and internal data. The *nowait* parameter is a choice of M_WAIT / M_DONTWAIT from caller. The *type* parameter is an mbuf type.

m_devget(*char *buf*, *int totlen*, *int off0*, *struct ifnet *ifp*, *void (*copy)* *__P((const void *, void *, size_t))*)
Copies *len* bytes from device local memory into mbufs using copy routine *copy*. If parameter *off* is non-zero, the packet is supposed to be trailer-encapsulated and *off* bytes plus the type and length fields will be skipped before copying. Returns the top of the mbuf chain it created.

m_copym(*struct mbuf *m*, *int off0*, *int len*, *int wait*)
Creates a copy of an mbuf chain starting *off0* bytes from the beginning, continuing for *len* bytes. If the *len* requested is M_COPYALL, the complete mbuf chain will be copied. The *wait* parameter is a choice of M_WAIT / M_DONTWAIT from caller.

m_copypacket(*struct mbuf *m*, *int how*)
Copies an entire packet, including header (which must be present). This function is an optimization of the common case m_copym(m, 0, M_COPYALL, *how*).

m_copydata(*struct mbuf *m*, *int off*, *int len*, *caddr_t cp*)
Copies *len* bytes data from mbuf chain *m* into the buffer *cp*, starting *off* bytes from the beginning.

m_copyback(*struct mbuf *m0*, *int off*, *int len*, *caddr_t cp*)
Copies *len* bytes data from buffer *cp* back into the mbuf chain *m0*, starting *off* bytes from the beginning, extending the mbuf chain if necessary.

m_cat(*struct mbuf *m*, *struct mbuf *n*)
Concatenates mbuf chain *n* to *m*. Both chains must be of the same type; packet headers will *not* be updated if present.

m_prepend(*struct mbuf *m*, *int len*, *int how*)
Lesser-used path for **M_PREPEND**(): allocates new mbuf *m* of size *len* to prepend to the chain, copying junk along. The *how* parameter is a choice of M_WAIT / M_DONTWAIT from caller.

m_pullup(*struct mbuf *m*, *int len*)
Rearranges an mbuf chain so that *len* bytes are contiguous and in the data area of an mbuf (so that **mtod**() will work for a structure of size *len*). Returns the resulting mbuf chain on success, frees it and returns NULL on failure. If there is room, it will add up to max_protohdr - *len* extra bytes to the contiguous region to possibly avoid being called again.

m_split(*struct mbuf *m0*, *int len0*, *int wait*)
Partitions an mbuf chain in two pieces, returning the tail, which is all but the first *len0* bytes. In case of failure, it returns NULL and attempts to restore the chain to its original state.

m_adj(*struct mbuf *mp*, *int req_len*)
Shaves off *req_len* bytes from head or tail of the (valid) data area. If *req_len* is greater than zero, front bytes are being shaved off, if it's smaller, from the back (and if it is zero, the mbuf will stay bearded). This function does not move data in any way, but is used to manipulate the data area pointer and data length variable of the mbuf in a non-clobbering way.

m_free(*struct mbuf *m*)
Frees mbuf *m*.

m_freem(*struct mbuf *m*)
Frees the mbuf chain beginning with *m*. This function contains the elementary sanity check for a NULL pointer.

mtod(*struct mbuf *m*, *datatype*)
Returns a pointer to the data contained in the specified mbuf *m*, type-casted to the specified data type *datatype*. Implemented as a macro.

mtocl(*void *datapointer*)
Takes a *datapointer* within an mbuf cluster and returns the cluster index number of the mbuf owning the data. Avoid this; it may be deprecated in the future. Implemented as a macro.

cltom(*u_long clusternum*)
Takes an mbuf cluster index number *clusternum* and returns a pointer to the beginning of the cluster. Avoid this; it may be deprecated in the future. Implemented as a macro.

MGET(*struct mbuf *m*, *int how*, *int type*)
Allocates mbuf *m* and initializes it to contain internal data. See **m_get**(). Implemented as a macro.

MGETHDR(*struct mbuf *m*, *int how*, *int type*)
Allocates mbuf *m* and initializes it to contain a packet header. See **m_gethdr**(). Implemented as a macro.

MEXTMALLOC(*struct mbuf *m*, *int len*, *int how*)
Allocates external storage of size *len* for mbuf *m*. The *how* parameter is a choice of M_WAIT / M_DONTWAIT from caller. The flag M_EXT is set upon success. Implemented as a macro.

MEXTADD(*struct mbuf *m*, *caddr_t buf*, *int type*, *void (*free) __P((caddr_t, u_int, void *))*, *void *arg*)
Adds pre-allocated external storage *buf* to a normal mbuf *m*; the parameters *type*, *free* and *arg* describe the external storage. *type* describes the malloc(9) type of the storage, *free* is a free routine (if not the usual one), and *arg* is a possible argument to the free routine. The flag M_EXT is set upon success. Implemented as a macro.

MCLGET(*struct mbuf *m*, *int how*)
Allocates and adds an mbuf cluster to a normal mbuf *m*. The *how* parameter is a choice of M_WAIT / M_DONTWAIT from caller. The flag M_EXT is set upon success. Implemented as a macro.

M_COPY_PKTHDR(*struct mbuf *to*, *struct mbuf *from*)
Copies the mbuf pkthdr from mbuf *from* to mbuf *to*. *from* must have the type flag M_PKTHDR set, and *to* must be empty. Implemented as a macro.

M_ALIGN(*struct mbuf *m*, *int len*)
Sets the data pointer of a newly allocated mbuf *m* to *len* bytes from the end of the mbuf data area, so that *len* bytes of data written to the mbuf *m*, starting at the data pointer, will be aligned to the end of the data area. Implemented as a macro.

MH_ALIGN(*struct mbuf *m*, *int len*)
Sets the data pointer of a newly allocated packetheader mbuf *m* to *len* bytes from the end of the mbuf data area, so that *len* bytes of data written to the mbuf *m*, starting at the data pointer, will be aligned to the end of the data area. Implemented as a macro.

M_LEADINGSPACE(*struct mbuf *m*)
Returns the amount of space available before the current start of valid data in mbuf *m*. Implemented as a macro.

M_TRAILINGSPACE(*struct mbuf *m*)
Returns the amount of space available after the current end of valid data in mbuf *m*. Implemented as a macro.

M_PREPEND(*struct mbuf *m*, *int plen*, *int how*)
Prepends space of size *plen* to mbuf *m*. If a new mbuf must be allocated, *how* specifies whether to wait. If *how* is M_DONTWAIT and allocation fails, the original mbuf chain is freed and *m* is set to NULL. Implemented as a macro.

MCHTYPE(*struct mbuf *m*, *int type*)
Change mbuf *m* to new type *type*. Implemented as a macro.

MEXTREMOVE(*struct mbuf *m*)
Removes external storage from mbuf *m*. The flag M_EXT is removed. Implemented as a macro.

MFREE(*struct mbuf *m*, *struct mbuf *n*)
Frees a single mbuf *m* and places the successor, if any, in mbuf *n*. Implemented as a macro.

SEE ALSO

/usr/share/doc/smm/18.net, netstat(1), malloc(9)

AUTHORS

The original mbuf data structures were designed by Rob Gurwitz when he did the initial TCP/IP implementation at BBN.

Further extensions and enhancements were made by Bill Joy, Sam Leffler, and Mike Karels at CSRG.

Current implementation of external storage by Matt Thomas
<matt@3am-software.com> and Jason R. Thorpe <thorpej@NetBSD.ORG>.

FILES

The **mbuf** management functions are implemented within the file `sys/kern/uipc_mbuf.c`. Function prototypes, and the functions implemented as macros are located in `sys/sys/mbuf.h`. Both pathnames are relative to the root of the NetBSD source tree, `/usr/src`.