**NAME**

copymsg – copy a message

**SYNOPSIS**

**#include <sys/stream.h>**

**mblk_t *copymsg(mblk_t ***_mp_**);**

**INTERFACE LEVEL**

Architecture independent level 1 (DDI/DKI).

**ARGUMENTS**

_mp_ Pointer to the message to be copied.

**DESCRIPTION**

**copymsg( )** forms a new message by allocating new message blocks, and copying the contents of the message referred to by _mp_ (using the **copyb**(9F) function). It returns a pointer to the new message.

**RETURN VALUES**

If the copy is successful, **copymsg( )** returns a pointer to the new message. Otherwise, it returns a **NULL** pointer.

**CONTEXT**

**copymsg( )** can be called from user or interrupt context.

**EXAMPLES**

The routine **lctouc( )** converts all the lowercase ASCII characters in the message to uppercase. If the reference count is greater than one (line 8), then the message is shared, and must be copied before changing the contents of the data buffer. If the call to the **copymsg( )** function fails (line 9), return **NULL** (line 10), otherwise, free the original message (line 11). If the reference count was equal to **1**, the message can be modified. For each character (line 16) in each message block (line 15), if it is a lowercase letter, convert it to an uppercase letter (line 18). A pointer to the converted message is returned (line 21).

```
1  mblk_t *lctouc(mp)
2       mblk_t *mp;
3  {
4       mblk_t *cmp;
5       mblk_t *tmp;
6       unsigned char *cp;
7
8       if (mp->b_datap->db_ref > 1) {
9               if ((cmp = copymsg(mp)) == NULL)
10                      return (NULL);
11              freemsg(mp);
12      } else {
13              cmp = mp;
14      }
15      for (tmp = cmp; tmp; tmp = tmp->b_cont) {
16              for (cp = tmp->b_rptr; cp < tmp->b_wptr; cp++) {
17                      if ((*cp <= 'z') && (*cp >= 'a'))
18                              *cp -= 0x20;
19              }
20      }
```

```
        return(cmp);
}
```

**SEE ALSO**

**allocb**(9F), **copyb**(9F), **msgb**(9S)

*Writing Device Drivers*
*STREAMS Programming Guide*