INTRODUCTION TO FILE MANAGER MESSAGES

The types of messages which the file manager is programmed to accept are:

READ	1	read from file
WRITE	2	write to file
OPEN	3	open file
CLOSE	4	close file
EXEC	5	open file for execution
FORK	6	increment count on open files
DELCAP	7	delete capability
CREAT	8	create file
LINK	9	link to a file
UNLINK	10	remove link from file
MDATE	11	modify date of file
CHDIR	12	change directory
INIT	13	initialization message
MKNOD	14	make a node
CHMOD	15	change mode of file
CHOWN	16	change owner of file
SYNC	17	update file systems on secondary
STAT	18	get status of file
FSIZE	19	get size of file
FSTAT	20	get status of open file
SMOUNT	21	mount file system
SUMOUNT	22	unmount file system
MOVE	23	move file into contiguous area
ALLOC	24	allocate contiguous space for file
OPENI	25	open file by inode number
TRUNC	26	truncate file length to given value
NMCODE	27	get segment name

For messages sent to the file manager, the first four words of the body of the message must contain the capability and the user and group ID's. The rest of the message contains various arguments depending on the message type. The structure of a message to the file manager is given by:

```
struct {
        struct msghdr;    /* 6 word message header */
        struct cp_clist {
                int     cpm_num;        /* capability number */
                int     cpm_owner;      /* capability owner */
                int     cpm_cap;        /* capability */
        };
        char    fm_uid;             /* user ID */
        char    fm_gid;             /* group ID */
        int     fm_arg[];           /* list of arguments */
}
```

The capability owner *cpm_owner* and the capability value *cpm_cap* are actually filled in by the kernel EMT from the PCB of the process which sent the message. The capability list *p_clist[]* is a list of the valid "capabilities" which the process has. A capability is a two-word entry which is put in the PCB by the memory manager process. A process may be given a capability only by the "owner" of the capabili-

ty. Typically, upon opening a file, the file manager will send an "add capability" message to the memory manager process. The memory manager will bring the PCB into its address space, find an empty capability slot and put the owner (file manager process number (4)) into the owner field of the capability and the capability itself into the capability field. The capability for the file manager is encoded as follows: 8 bits for in-core inode, 2 bits for read/write permissions and 6 bits for inode usage value. When a read or write message is sent to the file manager, a capability must be specified; this is checked for valid access permissions on the file by the file manager. Upon closing a file, the file manager sends a "delete capability" message to the memory manager.

The *fm_uid* and *fm_gid* identifiers are used to determine the message sender's access privileges to particular files. For all messages, error codes are passed back to the sender in *msstat.* A system error is indicated by a −1 value of *msstat.* The meanings of the possible error codes from the file manager are the same as the standard UNIX error codes as explained in section II of this manual. If the message type sent is illegal, an error code is also returned. The additional error codes are:

63 EBADTPYE bad message type

In messages which require a file name to be specified, the file pathname (a null-terminated string) is copied into the body of the message. A pathname may be up to 64 characters long. The start of each pathname string is specified by a byte offset into the body of the message starting at fm_arg[0].

The individual manual sections describing messages to the file manager list the arguments required as input. Here argument 0 is stored in *fm_arg[0],* argument 1 in *fm_arg[1],* etc. Again the returned values also refer to values returned in *fm_arg[i]* where *i* specifies the argument number. All messages to the file manager require a capability to be specified. In all cases this is the capability which refers to the current working directory or to the particular file being accessed.