SetFPos Position file mark for random-access read/write

#include <Files.h> **File Manager**

OSErr	**SetFPos(***fRefNum, posMode, posOffset* **);**	
short	*fRefNum* ;	file reference, as obtained via **FSOpen**
short	*posMode* ;	1=from start, 2=from end, 3=from mark
long	*posOffset* ;	file offset, absolute pos depends on *posMode*
	returns	Error Code; 0=no error

SetFPos sets the position of the file mark (the file position at which the next read or write operation will start).

fRefNum is the reference number of an open file. See **FSOpen** and **OpenRF**.

posMode specifies the method by which the file pointer will be moved. The following constants are defined in Files.h:

fsAtMark	0	Remain at current mark (*posOffset* is ignored)
fsFromStart	1	Move to absolute file position in *posOffset*
fsFromLEOF	2	Move *posOffset* bytes from logical end of file
fsFromMark	3	Move *posOffset* bytes from current position

posOffset is a signed long integer (positive or negative); it identifies how far to move the file mark. The resulting absolute file position will depend upon the method specified by *posMode*.

Returns: an operating system Error Code. It will be one of:

noErr	(0)	No error
eofErr	(-39)	Attempt tp position past the end-of-file
extFSErr	(-58)	External file system
fnOpnErr	(-38)	File not open
ioErr	(-36)	I/O error
posErr	(-40)	Can't position to before start of file
rfNumErr	(-51)	Bad *fRefNum*

Notes: **SetFPos** (LSEEK to UNIX fans) is used in random-access disk operations to position the file mark to a specified position in order to read from or write to a selected position in the file. This function is not needed in sequential file I/O, since the file mark is updated automatically via **FSRead** and **FSWrite**.

The *posOffset* parameter may be positive or negative. If the combination of *posMode* and a positive *posOffset* would move past the end of the file, the mark is set to the EOF and eofErr is returned. If you attempt to position the file mark before the start of the file (i.e., while using a negative value in *posOffset*) posErr is returned.

If you wish to append records to the end of the file, simply use *posMode*=fsFromLEOF and *posOffset*=0 (see **FSWrite** for an example). If you need to seek beyond the end of the file, you can use **Allocate** or **SetEOF** to add empty space to the end of the file before using **SetFPos**.

Some examples:

```
    FSOpen( "\pHardDisk:MyFile", 0, &fRef );
```

```
FSRead( fRef, 20, myBuf );              /* mark now points to 21st byte */

SetFPos( fRef,fsFromStart,0 );          /* rewind to start of file */
SetFPos( fRef,fsFromLEOF,0 );           /* prepare to append to file */
SetFPos( fRef,fsFromMark,-10 );         /* rewind by 10 bytes */

newRecAddr = GetNameHashAddr( theName );  /* calc record address */
err = SetFPos( fRef, fsFromStart, newRecAddr );
if (err == eofErr) {                      /* file is too small */
     SetEOF( fRef,newRecAddr+REC_SIZE ); /* increase file size */
     SetFPos( fRef, fsFromStart, newRecAddr );
}
```

Note: The fsAtMark constant is meaningless here, but it is needed in the low-level **PBRead** and **PBWrite** functions.