SetEOF Increase or decrease the logical size of a file

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

OSErr	**SetEOF(***fRefNum*, *newEOF* **);**	
short	*fRefNum* ;	file reference, as obtained via **FSOpen**
long	*newEOF* ;	desired file size, in bytes
	returns	Error Code; 0=no error

Use **SetEOF** to change the size of a file to any arbitrary length. Disk blocks are allocated or released to accommodate the request.

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

newEOF is the desired new size of the file, in bytes.

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

noErr	(0)	No error
dskFulErr	(-34)	Disk full (partial allocation made)
extFSErr	(-58)	External file system
fLckdErr	(-45)	File is locked
fnOpnErr	(-38)	File not open
ioErr	(-36)	I/O error
rfNumErr	(-51)	Bad *fRefNum*
vLckdErr	(-46)	Volume is locked
wPrErr	(-44)	Diskette is write-protected
wrPermErr	(-61)	Write permissions error

Notes: If *newEOF* is larger than the current file size (see **GetEOF**), the file size is increased by allocating additional disk blocks to the physical EOF (if needed). If there is not enough available disk space to satisfy the entire request the dskFulErr is returned and no new space is allocated.

You can also use **Allocate** to increase the size of a file. The **PBAllocContig** function may be preferable since it attempts to allocate contiguous blocks (for best read/write performance).

If *newEOF* is smaller than the current size and if the new size is small enough to fit in fewer allocation blocks, disk blocks will be released as the file is truncated. For instance,

```
SetEOF( fRef, 0 );
```

sets the logical end-of-file to 0 and releases all the disk blocks allocated to the file (thus, freeing up space on the disk).