PBSetEOF Set the logical file size of an open file

#include <Files.h> **File Manager (PBxxx)**

```
OSErr       PBSetEOF(pb, async);
ParmBlkPtr  pb ;        address of a 50-byte IOParam structure
Boolean     async ;     0=await completion; 1=immediate return
            returns     Error Code; 0=no error
```

PBSetEOF lets you expand or truncate the logical size of an open file. The file must be open with read/write permission.

pb is the address of a 50-byte IOParam structure. The relevant fields are as follows:

Out-In	Name	Type	Size	Offset	Description
->	ioCompletion	ProcPtr	4	12	Completion routine address (if *async* =TRUE)
->	ioRefNum	short	2	24	File reference number
->	ioMisc	Ptr	4	28	Desired logical file size, in bytes
<-	ioResult	OSErr	2	16	Error Code (0=no error, 1=not done yet)

async is a Boolean value. Use FALSE for normal (synchronous) operation or TRUE to enqueue the request and resume control immediately. See Async I/O.

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

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

Notes: Place the desired size in ioMisc. Since that structure field is declared as a type Ptr, you will need to coerce the desired size from a long integer to a Ptr:

```
IOParam   pb;
long      newSize;

newSize = 1000L;
pb.ioRefNum = myFileRef;
pb.ioMisc = (Ptr)newSize;
err = PBSetEOF( &pb, FALSE );
```

If you set the logical EOF to a value larger than its physical EOF, additional disk blocks will be allocated to the file. If there is not enough space on the disk to satisfy the request, *no change is made* and the function returns dskFulErr.

Using a smaller size will truncate the file. The resulting physical EOF will be the value of ioMisc, rounded up to the next higher block size. Use ioMisc=0 to truncate all data from the open fork of the file and release all

of its storage on the volume.

You can use **PBAllocate** or **PBAllocContig** to increase the physical size of the file, without affecting the logical EOF. Use **PBGetFInfo** to obtain the current value of the physical EOF.

Note that since sequential write operations (starting at the end of the file) increase the EOF automatically, this function is rarely needed.