PBGetFCBInfo Obtain information from open file control blocks

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

```
OSErr      PBGetFCBInfo(pb, async);
FCBPBPtr   pb ;      address of a 62-byte FCBPBRec structure
Boolean    async ;   0=await completion; 1=immediate return
           returns   Error Code; 0=no error
```

PBGetFCBInfo returns information from an FCB (file control block; a potpourri of data maintained about each open file). You can use this to index through all FCBs.

pb is the address of a 62-byte FCBPBRec structure. The relevant fields are as follows:

Out-In	Name	Type	Size	Offset	Description
->	ioCompletion	ProcPtr	4	12	Completion routine address (if *async* =TRUE)
->	ioFCBIndx	short	2	28	Index (or 0 if not indexing)
<->	ioVRefNum	short	2	22	Volume or working directory (0=all open files)
<->	ioRefNum	short	2	24	File reference number (from **PBOpen**)
<-	ioResult	OSErr	2	16	Error Code (0=no error, 1=not done yet)
<-	ioNamePtr	StringPtr	4	18	Address of buffer to receive 32-byte filename
<-	ioFCBFlNm	long	4	32	Unique 'hard' file number
<-	ioFCBFlags	short	2	36	Flags (bit 8=has write permission, bit 9=resource fork, bit 15=dirty)
<-	ioFCBStBlk	short	2	38	First allocation block
<-	ioFCBEOF	long	4	40	Logical EOF (file size, in bytes)
<-	ioFCBPLen	long	4	44	Physical EOF
<-	ioFCBCrPs	long	4	48	Current file position (mark)
<-	ioFCBVRefNum	short	2	52	'Hard' volume reference number
<-	ioFCBClpSiz	long	4	54	File clump size (minimum allocation unit)
<-	ioFCBParID	long	4	58	ID of directory in which file resides

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
extFSErr	(-58)	External file system
fnOpnErr	(-38)	File not open (indexed past last FCB)
fsDSIntErr	(-127)	Internal file system error
nsvErr	(-35)	No such volume
paramErr	(-50)	Can't get info on file server volume

Notes: There's little reason to need this command since other calls such as **PBGetEOF**, **PBGetFPos**, and **PBGetFInfo** return most of it.

PBGetFCBInfo is not supported by the pre-HFS File Manager. You can get to this information (and other fields of the FCB) by examining the global variable FCBSPtr (at 0x034E) which contains the address of the 16-bit length of the FCB buffer. The first FCB begins directly after that length word. Examine FSFCBLen (at 0x03F6) and use that increment to point to succeeding FCBs. Note: if FSFCBLen=0, you can be sure HFS is not present and you can assume an FCB length of 30 bytes.

To get information about a single file, set ioFCBIndx=0, and put the file's

reference number (as obtained from **PBOpen**) in ioRefNum. Be sure that ioNamePtr points to a buffer to receive the filename (or is NIL, if you don't care).

To examine all files, set ioNamePtr to the address of a 32-byte minimum buffer to receive the filename. Set ioVRefNum to identify the volume of interest (drive number, 'hard' volume ID, or working directory reference). If you want the big picture, set ioVRefNum to 0; that will index through all volumes. Finally, set ioFCBIndx to 1 and iterate it through successively higher numbers until the call returns an error.

If your compiler's header file defines ioFCBFlags as a 2-byte value, use masks of 0x0100, 0x0200, and 0x0800 to read the flags. IM IV Pg 180 identifies this field as a byte, and therefore lays out different masks.