PBHGetFInfo Query file date/time, attributes, type... (HFS only)

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

```
OSErr       PBHGetFInfo(pb, async);
HParmBlkPtr pb ;        address of an 80-byte HFileParam structure
Boolean     async ;     0=await completion; 1=immediate return
            returns     Error Code; 0=no error
```

PBHGetFInfo obtains a variety of information about one file or all files in an HFS directory. It does not return information about subdirectories.

pb is the address of an 80-byte HFileParam structure (or a fileParam.member of an HParamBlockRec union). The relevant fields are as follows:

Out-In	Name	Type	Size	Offset	Description
–>	ioCompletion	ProcPtr	4	12	Completion routine address (if *async* =TRUE)
–>	ioVRefNum	short	2	22	Volume, drive, or working directory reference
–>	ioFVersNum	SignedByte	1	26	Version (best to use 0)
–>	ioFDirIndex	short	2	28	Index (use 0 if not indexing)
<->	ioNamePtr	StringPtr	4	18	Entry: Full/partial path/filename (if not indexing) Exit: Receives one-element filename (if indexing)
<->	ioDirID	long	4	48	Entry: "hard" directory ID (0=use ioVRefNum) Return: 'Hard" file number
<-	ioResult	OSErr	2	16	Error Code (0=no error, 1=not done yet)
<-	ioFRefNum	short	2	24	File access path reference number
<-	ioFlAttrib	SignedByte	1	30	File Attribute bits (locked, directory, etc.)
<-	ioFlVersNum	SignedByte	1	31	File version (best to use 0)
<-	ioFlFndrInfo	FInfo	16	32	(File type, creator, flags, icon position, etc.)
<-	ioFlStBlk	short	2	52	First allocation block of data fork
<-	ioFlLgLen	long	4	54	Logical end-of-file of data fork
<-	ioFlPyLen	long	4	68	Physical end-of-file of data fork
<-	ioFlRStBlk	short	2	62	First allocation block of resource fork
<-	ioFlRLgLen	long	4	64	Logical end-of-file of resource fork
<-	ioFlRPyLen	long	4	68	Physical end-of-file of resource fork
<-	ioFlCrDat	long	4	72	Date/Time of creation (seconds since 1/1/1904)
<-	ioFlMdDat	long	4	76	Date/Time of last modification

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
bdNamErr	(-37)	Bad name
dirNFErr	(-120)	Directory not found
extFSErr	(-58)	External file system
fnfErr	(-43)	File not found
ioErr	(-36)	I/O error
nsvErr	(-35)	No such volume
paramErr	(-50)	No default volume

Notes: **PBHGetFInfo** is identical to **PBGetFInfo** except that the field at parameter block offset 48 is called ioDirID and may be used to specify the directory to peruse.

On entry, ioDirID may contain 0, indicating that ioVRefNum identifies the

directory of interest, or it may contain a 32-bit "hard" directory ID (as found in CurDirStore). In the latter case, ioVRefNum is taken as a "hard" volume ID (never a working directory reference).

When indexing, ioDirID gets overwritten on each iteration. You should reinitialize it before each call to **PBHGetFInfo**.

You may prefer to use **PBGetCatInfo**, since it returns information about directories as well as regular files (an example under that topic illustrates recursive directory searching). The high-level **GetFInfo** is easier to use and may satisfy your needs. See **PBGetFInfo** for related usage guidelines.