PBHSetFInfo Change file date/time, type, etc. (HFS only)

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

```
OSErr       PBHSetFInfo(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
```

PBHSetFInfo lets you modify selected fields of a file's information structure. It is similar to **PBSetFInfo**, but it allows you to specify a file's directory using a "hard" directory ID.

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)
->	ioNamePtr	StringPtr	4	18	Full or partial path/filename
->	ioVRefNum	short	2	22	Volume, drive, or working directory reference
->	ioFlFndrInfo	FInfo	16	32	(File type, creator, flags, icon position, etc.)
->	ioDirID	long	4	48	"hard" directory ID (0=use ioVRefNum)
->	ioFlCrDat	long	4	72	Date/Time of creation (seconds since 1/1/1904)
->	ioFlMdDat	long	4	76	Date/Time of last modification
<-	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
dirNFErr	(-120)	Directory not found
extFSErr	(-58)	External file system
fLckdErr	(-45)	File is locked
fnfErr	(-43)	File not found
ioErr	(-36)	I/O error
nsvErr	(-35)	No such volume
vLckdErr	(-46)	Volume is locked
wPrErr	(-44)	Diskette is write-protected

Notes: Before calling **PBHSetFInfo**, you should always call **PBHGetFInfo**, in order to fill the parameter block with current values of all the fields you do not wish to change.

As with other **PBH**xxx calls, you can specify the file of interest in one of several ways:

- ioDirID is 0. ioNamePtr is a partial pathname or a one-element filename. ioVRefNum identifies the parent directory of the start of the path - either a working directory reference or a "hard" volume number (or 0 to indicate the current default volume). You would use this option after calling Standard File.
- ioDirID is 0. ioNamePtr is a complete pathname (starting at the root) and ioVRefNum is a "hard" volume number or disk number (or 0=default).
- ioDirID is a "hard" directory number. ioNamePtr is a pathname leading

from that directory to the file. ioVRefNum must be a "hard" volume number (or 0=default) and NOT a working directory number.

Since **PBHSetFInfo** is always called after **PBHGetFInfo**, pay attention to the ioDirID field (since **PBHGetFInfo** fills this with a directory ID). Also be sure to flush the volume to disk (**PBFlushVol**); otherwise, the Finder might not notice the changed information (it may not anyway - the Finder appears to do some internal caching).

The **PBSetCatInfo** is more flexible since it lets you change information maintained about directories as well as files. The high-level **SetFInfo** may be all you need - it only lets you change the Finder information.

The following example changes a file's Finder flags to make its icon invisible from the Finder.

Example

```
#include <Files.h>

HFileParam      hfpb;
short           rc;
short           myWorkDir;    /* assume prior call to PBOpenWD */

hfpb.ioNamePtr = (StringPtr) "\pSecretFile";
hfpb.ioVRefNum = myWorkDir;
hfpb.ioDirID=0;                         /* not used this time */
rc = PBHGetFInfo( &hfpb, FALSE );
if ( rc ) { /* . . . handle the error . . . */ }

hfpb.ioFlFndrInfo.fdFlags |= fInvisible;      /* bit 14=hide the file */

hfpb.ioDirID=0;                     /* was overwritten by PBHGetFInfo() */
rc = PBHSetFInfo( &hfpb, FALSE );
if ( rc ) { /* . . . handle the error . . . */ }
PBFlushVol(  &hfpb, FALSE );      /* else change might not be noticed */
```