PBHOpen Open file data fork (HFS only)

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

OSErr	**PBHOpen(***pb*, *async* **);**	
HParmBlkPtr	*pb* ;	address of a HParamBlockRec union
Boolean	*async* ;	0=await completion; 1=immediate return
	returns	Error Code; 0=no error

PBHOpen opens the data fork of a file, enabling I/O operations. It is the same as **PBOpen**, except that you can specify a "hard" directory ID in the parameter block.

pb is the address of an 122-byte HParamBlockRec union. This call uses members of two different structures (see the example). The following fields are relevant:

Out-In	Name	Type	Size	Offset	Struct	Description
–>	ioCompletion	ProcPtr	4	12	ioParam	Completion rtn address (only used if *async* = TRUE)
–>	ioNamePtr	StringPtr	4	18	ioParam	Address of full or partial path/filename
–>	ioVRefNum	short	2	22	ioParam	Volume, drive, or directory ref
–>	ioPermssn	SignedByte	1	27	ioParam	File Permission (1=read, 2=write...)
–>	ioMisc	Ptr	4	28	ioParam	Address of 522-byte buf (0=use vol buf)
–>	ioDirID	long	4	48	fileParam	Directory ID (0=use ioVRefNum)
<-	ioResult	OSErr	2	16	ioParam	Error Code (0=no err, 1=not done yet)
<-	ioRefNum	short	2	24	ioParam	File reference number

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
opWrErr	(-49)	File already open for writing
permErr	(-54)	Attempt to open locked file for writing
tmfoErr	(-42)	Too many files open

Notes: **PBHOpen** opens an access path for the file identified by ioParam.ioNamePtr and ioVRefNum, as in **PBOpen**. However, if a directory ID is used in fileParam.ioDirID, (such as one obtained via **PBDirCreate** or the global variable CurDirStore), then it will override any working-directory reference you use in ioVRefNum.

The ioParam.ioPermssn field specifies read-only, read/write, and sharing options. In most cases, you can simply set the permission parameter to fsCurPerm. Some applications request fsRdWrPerm, to ensure that they can both read and write to a file. The constants that can be passed in this field are the following:

fsCurPerm	exclusive read/write permission if it is available; otherwise, exclusive read, if that is available
fsRdPerm	exclusive read permission
fsWrPerm	exclusive write permission
fsRdWrPerm	exclusive read/write permission
fsRdWrShPerm	shared read/write permission

In shared environments, permission requests are translated into the "deny-mode" permissions defined by AppleShare.

Set ioParam.ioMisc to 0 for normal I/O buffering (via the volume buffer), or set it to point to a 522-byte area of memory.

Here's what a typical call might look like:

```
#include <Files.h>

HParamBlockRec pb;          // create a 122-byte parm block
OSErr rc;

pb.ioParam.ioNamePtr = (StringPtr)"\pMyFile";
pb.ioParam.ioPermssn = fsRdPerm; // read-only
pb.ioParam.ioMisc = 0;            // use volume buffer
pb.fileParam.ioDirID = CurDirStore;   // dir opened by SFPkg
rc = PBHOpen( &pb, FALSE );
```

You might prefer to use a pointer to the union, e.g.:

```
HParamBlockRec    pb;     /* a 122-byte parm block */
HParmBlkPtr       pbp;    /* note spelling convention */


pbp = &pb;
pbp->ioParam.ioNamePtr = (StringPtr)"\pMyFile";
pbp->ioParam.ioPermssn = fsRdPerm;

    /* (etc.) */
```