PBOpenWD Create/get reference number of working directory

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

```
OSErr     PBOpenWD(pb, async);
WDPBPtr   pb ;      address of a 52-byte WDPBRec structure
Boolean   async ;   0=await completion; 1=immediate return
          returns   Error Code; 0=no error
```

PBOpenWD creates a working directory control block to identify a specified directory, or if one already exists, it obtains the existing reference number. A working directory number can be used whenever a volume reference number is required.

pb is the address of a 52-byte WDPBRec structure. 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	Address of full or partial path/filename
->	ioWDProcID	long	4	28	Working directory user ref (app's signature or 0)
->	ioWDDirID	long	4	48	Working directory's directory ID
<->	ioVRefNum	short	2	22	Entry: Volume, drive, or working dir reference Return: Working directory reference number
<-	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
tmwdoErr	(-121)	Too many working directories open

Notes: "Working directories" exist as a way to maintain compatibility with the flat file system of the original Macintosh. In effect, a working directory reference number is just a 16-bit alias for a 32-bit "hard" directory ID and a 'real' volume reference. When the File Manager sees a working directory reference in a volume reference number (e.g., ioVRefNum of any parameter block), it looks up the real directory number and uses that value in finding files.

Note: This notion of a working directory is very different from that of a 'current default directory,' as MS-DOS and especially UNIX programmers might assume. See **SetVol** for details on setting the default volume.

The Standard File Package calls **PBOpenWD** each time the user selects a directory (either by opening a folder or selecting from the drop-down list). If the user then opens a file, the resulting working directory reference number is returned in vRefNum. Thus, if you use Standard File for file selection, you may never need **PBOpenWD**.

Note: You can eyeball a VRefNum and see what it is: large negative numbers such as -32456 (0x8138) generally are used for working directories; small negative numbers such as -1 (0xFFFF) are most often real volume numbers. See HFS Notes for more information.

The system maintains a limited number of slots for tracking working directories. It is wise to close WDs (via **PBCloseWD**) as soon as possible.

The ioWDProcID field lets you personalize the working directory control block. If you put a non-0 value in here, you can use **PBGetWDInfo** to index through only those WDs with a selected ioWDProcID value. Normal usage is to store your application signature in that field, or use 0 if you don't care.

As with **PBH**xxx calls, you can identify the directory via its full pathname and put 0 in ioWDDirID, or you can identify a directory with ioWDDirID and select a subdirectory via ioNamePtr (or set it to NIL for the ioWDDirID directory itself).

Example

```
#include <Files.h>

WDPBRec     wdpb;
short       rc;

wdpb.ioNamePtr="\pHardDisk:Ltrs:Current";
wdpb.ioWDDirID=0;                       /* using 0 since name.... */
wdpb.ioVRefNum=0;                   /* ... identifies fully */
wdpb.ioWDProcID='DRYL';             /* Add indivual identification */

rc = PBOpenWD( &wdpb, FALSE );
if ( rc ) { /* . . . handle the error . . . */ }
```