FSOpen Open the data fork of an existing file

#include <Files.h> **File Manager**

```
OSErr     FSOpen(fileName, vRefNum, fRefNum );
Str255    fileName ;     address of length-prefixed full or partial name
short     vRefNum ;      volume or directory reference number
short     *fRefNum ;     receives file reference number
          returns        Error Code; 0=no error
```

FSOpen opens the data fork of an existing file for read/write or read-only access (depending upon whether or not the file is locked).

fileName is the address of a length-prefixed, pascal-style string containing the name of the file to be opened. It may be a partial or full pathname, depending upon the value of *vRefNum* . If it is a fully-qualified pathname, *vRefNum* is ignored.

vRefNum is the reference number of the volume or directory containing *fileName*. It may be one of:

volume ref num A "hard" volume number; i.e., the refNum of the root directory on the volume. In this case, *fileName* must specify the rest of the name, including all subdirectories leading from the root to the file.

directory ref num A directory reference number as returned by **PBOpenWD** and **SFPutFile** (et.al.) . In this case, *fileName* can be a simple one-name file in that directory, or a multiple-name pathname including directories leading from *vRefNum* to the file.

0 This specifies the current default volume/directory, as set via **SetVol** or **PBHSetVol**. In this case, *fileName* can be a full or partial pathname as above.

fRefNum is the address of a 16-bit short. Upon return, it will contain the file reference number (also called the access path number) of the file. This value is used in all subsequent operations on the open file.

Returns: an operating system Error Code. It will be one of:

noErr	(0)	No error
nsvErr	(-35)	No such volume
ioErr	(-36)	I/O error
bdNamErr	(-37)	Bad name
tmfoErr	(-42)	Too many files open
fnfErr	(-43)	File not found
opWrErr	(-49)	File already open for writing
extFSErr	(-58)	External file system

Notes: The file must exist (use **Create** to create a new file). If *fileName* is not found in the volume/directory specified, then the File Manager searches the directory containing the currently-open System File (its volume number is in the global variable BootDrive at 0x0210). If the file is not there, then the File Manager searches the System Folder (actually, it searches the directory whose "hard" ID is in the first long word in the ioVFndrInfo field of the HVolumeParam structure returned by a call to **PBHGetVInfo**).

When found, the file is opened with whatever access permission is allowed; that is, if a file is locked, it is opened "read-only" and any subsequent attempt to write to the file will fail with an error. Use **RstFLock** to unlock a file.

If the file is locked (read-only) and has already been opened, a new access path is created with its own buffer and file mark. If the file is read/write and is already open, then this call will return opWrErr, and *fRefNum* will be set to the existing file reference number. You may use the low-level **PBOpen** if you wish to open the file with read-only access or if you need to open multiple access buffers for a file.

You can use various combinations of *vRefNum* and a full or partial *fileName*, as long as the combination specifies exactly where to find the file. See **SetVol** for examples that select a default directory, and use *vRefNum* =0 in the call to **FSOpen**.

FSOpen is typically followed by calls to **FSRead**, **FSWrite**, **SetFPos**, **FlushVol** etc. Use **FSClose** when you are finished with the file; but it will be closed automatically when the application terminates (**ExitToShell**) or the volume is unmounted (**UnmountVol**) or taken offline (**PBOffLine**).

Use **Create** to create a new empty file before trying to open it. Use **OpenRF** to access the resource fork of the file (e.g., in a file-copy operation).

The most common way to obtain *fileName* and *vRefNum* is by using the Standard File Package, as illustrated in this example.

Example

```
#include <Files.h>
#include <StandardFile.h>

SFReply     tr;
short       rc, fRefNum;
Point       where;

where.h=100; where.v=50;    /* where the  Standard File dialog window goes */

SFGetFile( where, "\pSelect a file", 0, -1, 0, 0, &tr );

if ( tr.good ) {
    rc = FSOpen( tr.fName, tr.vRefNum, &fRefNum );
    if ( rc ) { /* . . . handle the error . . . */ }
    MyReadFile( fRefNum );                  /* read the file */
    FSClose( fRefNum );                     /* close it */
}
```

If you know where the file is (in the same folder as the application or in the System Folder, for example), and you don't want to use the Standard File Package, you could call **FSOpen** followed by a call to **GetVRefNum**. This

will get you the vRefNum of the file, which is the same thing that the Standard File Package would have returned.