HSetFLock Lock a file (prevent changes, deletion, renaming, etc.)

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

```
OSErr    HSetFLock(vRefNum, dirID, fileName );
short    vRefNum ;    volume or working directory reference
long     dirID;       volume or directory ID
Str255   fileName ;   address of length-prefixed full or partial name
         returns      Error Code; 0=no error
```

HSetFLock locks a file. This prevents programs from modifying it in any way - deleting, renaming, or writing to either its data or resource fork. It is similar to **SetFLock** except that it takes a vRefNum/dirID combination instead of just a vRefNum.

vRefNum is the reference number of the volume or working directory that contains the file or directory *fileName*. Use 0 to specify the default volume.

dirID is the ID of the directory where the file resides.

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

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

noErr	(0)	No error
extFSErr	(-58)	External file system
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: This sets the file's "lock" flag (as found in the ioFlAttrib field of the FileParam structure) and notifies the system of the change (Note: if you change this bit directly, as with **PBSetCatInfo**, the change may not be noticed by the Finder until the file's folder is closed and reopened or the system is restarted).

This prevents programs from deleting (**FSDelete**), renaming (**Rename**), or writing (**FSWrite**) to the file. Any attempt to open the file (**FSOpen**) for read/write access will fail. Of course, any process can unlock the file (via **RstFLock**) if it wants such access.

This has no affect on currently-open access paths. Thus, you can open a file for writing, then lock it to prevent other concurrent processes from writing to it. Afterward, use **RstFLock** to unlock the file.

You can lock/unlock an entire volume via **PBSetVInfo** or lock a selected portion of an open file via **PBLockRange**. Use **PBGetFInfo** to see if a file is currently locked (ioFlAttrib bit 1 is set).

Be sure to call **FlushVol** to make sure that the change is written to the disk in a timely manner.