Macintosh Technical Notes

#44: HFS Compatibility

See also: The File Manager

Written by: Jim Friedlander October 9, 1985
Modified by: Scott Knaster December 5, 1985
Jim Friedlander
Updated: March 1, 1988

This technical note tells you how to make sure that your applications run under the Hierarchical File System (HFS).

The Hierarchical File System (HFS) provides fast, efficient management of larger volumes than the original Macintosh File System (MFS). Since HFS is hierarchical, HFS folders have a meaning different from MFS folders. In MFS, a folder has only graphical significance—it is only used by the Finder as a means of visually grouping files. The MFS directory structure is actually flat (all files are at the 'root' level). Under HFS, a folder is a directory that can contain files and other directories.

A folder is accessed by use of a `WDRefNum` (Working Directory reference number). Calls that return a `vRefNum` when running under MFS may return a `WDRefNum` when running under HFS. You may use a `WDRefNum` wherever a `vRefNum` may be used.

In order to provide for compatibility with software written for MFS, the HFS calls that open files search both the default directory and the directory that contains the System and the Finder (HFS marks this last directory so it always knows where to look for the System and the Finder).

Your goal should be to write programs that are file system independent. Your programs should not only be able to access files on other volumes, but also files that are in other directories. Accomplishing this is not difficult—most applications that were written for MFS work correctly under HFS. If you find that your current applications do not run correctly under HFS, you should check to see if you are doing any of the following five things:

Are you using Standard File?

This is very important to ensure that your application will run correctly under HFS. HFS uses an extended Standard File, which allows the user to select from files in different directories. This increased functionality was implemented without changing Standard File's external specification—the only difference is that `SFReply.vRefNum` can now be a `WDRefNum`. Please note that using Standard File's dialog hook and filter procs or adding controls of your own will not cause compatibility problems with HFS.

Existing applications that use Standard File properly run without modification under HFS. Applications that take the `SFReply.vRefNum` and convert that to a volume name, then append it to `SFReply.fName` (as in #2 below) do not function correctly under HFS—the user can only open files in the root directory. If you call `Open` with `SFReply.vRefNum` and

`SFReply.fName`, everything will work correctly. Remember, `SFReply.vRefNum` may be a `WDRefNum` . Using Standard File will virtually guarantee that your application will be compatible with MFS, HFS, and future file systems.

Are you concatenating volume names to file names, i.e. using file names of the form `VOLUME:fileName`?

Applications that do this do not work correctly under HFS (in fact, they do not even run correctly under MFS). Instead of this, use a `vRefNum` to access a volume or a directory. Fully qualified pathnames (such as `volume:folder1:folder2:filename`) work correctly, but we don't recommend that you use them. Please don't ever make a user type in a full pathname!

Are you searching directories for files using a loop such as `FOR index:= 1 to ioVNmFls DO ...` where `ioVNmFls` was returned from a `PBGetVinfo` call?

This technique should not be used. Instead, use repeated calls to `PBGetFInfo` using `ioFDirIndex` until `fnfErr` is returned. Indexed calls to `PBGetFInfo` will return files in the directory specified by the `vRefNum` that you put in the parameter block.

Are you assuming that a `vRefNum` will actually refer to a volume?

A `vRefNum` can now be a `WDRefNum`. A `WDRefNum` indicates which working directory (folder) a file is in, not which volume the file is on. Don't think of a `vRefNum` as a way to access a volume, but rather as a means of telling the file system where to find a file.

Are you walking through the VCB queue?

You should let us do the walking for you. Using indexed calls to `PBGetVInfo` will allow you to get information about any mounted volume. You shouldn't walk through the VCB queue because it changed for HFS and might change in the future. The routines that we supply will correctly access information in the VCB queue.

Are you using the file system's "`IMMED`" bit? (assembly language only)

Inside Macintosh describes bit 9 of the trap word as the immediate bit. In fact, setting this bit under MFS did not work as documented; it did not have the desired effect of bypassing the file I/O queue. Under HFS, this bit is used; it distinguishes HFS varieties of calls from MFS varieties. For example, the `PBOpen` call has this bit clear; `PBHOpen` has it set. Therefore, you must be sure that your file system calls do not use this bit as the immediate bit.