Technical Note QT01
Dependent Files

CONTENTS

This Technical Note describes how to identify and work with dependent files, which are files that reference or are referenced by other files. It also documents QuickTime's dependent file format.

[Mar 01 1993]

Introduction

Dependent files are files that reference other files or are referenced themselves by other files. In this note, methods of working with QuickTime dependent files, as well as QuickTime's dependent file format are documented.

This note discusses issues mainly applicable to file management software. Most applications rely on the Finder for file management techniques such as copying, deleting, and so on. If your application does not delete or copy files with signatures other than your own and does not work with QuickTime files, then this note probably does not concern you. On the other hand, your application may already create dependent files, and you may wish to adopt QuickTime's method of tracking them.

Back to top

Dependent File Overview

A QuickTime movie file may reference more than one file. In a common scenario, the movie's data may be stored in one file while the movie's resource may be stored in another. In fact, a QuickTime movie file may reference data stored in several files. For example, sound might be stored in one file, the video in another, and the movie resource itself in yet another. Logically, though, these files belong together and, thus, dependent files were created.

Dependent files use customized aliases to refer to the other files.

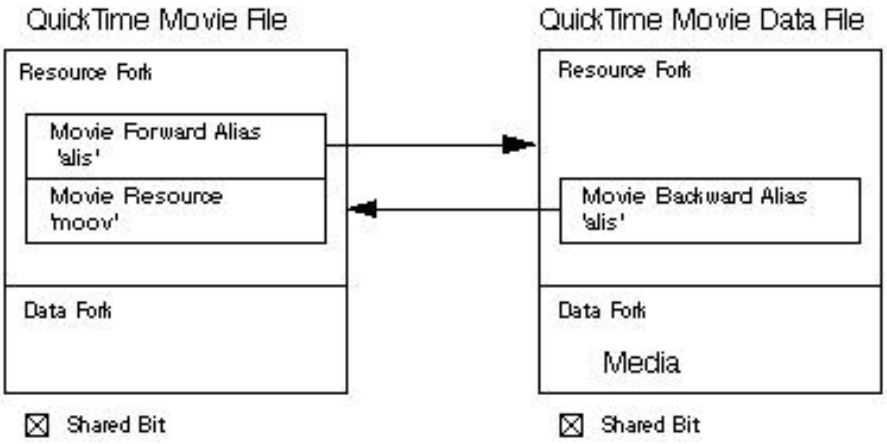

Figure 1. Diagram of Two Dependent Files

Two types of customized aliases are used: forward aliases and backward aliases. Files that reference other files contain a forward alias to the referenced file. Files that are referenced by another file contain a backward alias to the referencing file. To enable quick identification of a dependent file, the "shared" bit of the file's Finder Information is set (*Inside Macintosh* Volume VI, page 9-37).

A dependent file is a file that contains a dependent alias (either a backward or forward alias) and whose shared bit is set. If both conditions do not exist, then the file is not dependent.

New Use of Shared Bit

The shared bit is bit number 6 of the `fdFldr` field of the file's `FInfo` record. The following sample code demonstrates the proper method of identifying a dependent file.

```
OSErr FileIsDependent(FSSpec *anyFSSpec,Boolean *isShared)
{
    #define sharedBit (1<<6)
    OSErr    err;
    FInfo    fileFInfo;

    err = FSpGetFInfo(anyFSSpec,&fileFInfo);
    if (err) return err;
    if ((fileFInfo.fdFlags & sharedBit) && (fileFInfo.fdType != 'APPL') )
        *isShared = true;
    else
        *isShared = false;
    return noErr;
}
```

Previously, the shared bit applied only to applications that could be used on a network volume by multiple users. The new use does not obscure its previous use. If the file is an application and the bit is set, then it is available to multiple users and its old meaning is retained. If the file is not an application, and the bit is set then the file may depend on other files.

Dependent Alias Format

Aliases can be customized in two ways: the `userType` field can be modified and custom data can be added after the alias' private data structure (*Inside Macintosh* Volume VI, page 27-12).

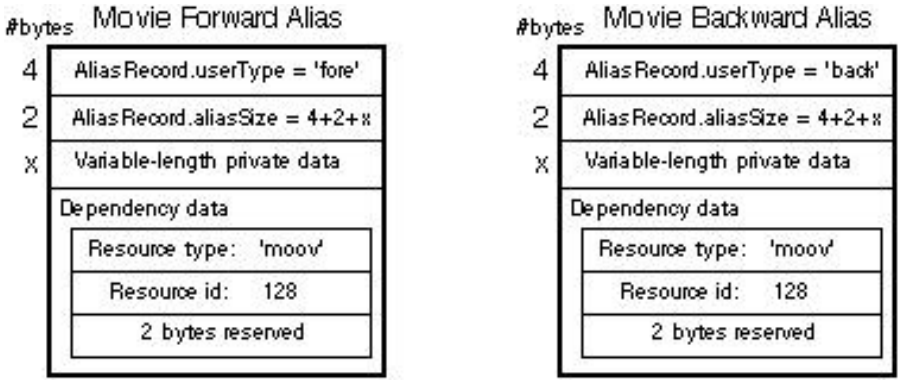

Figure 2. Diagram: Movie Forward Alias & Movie Backward Alias From Fig. 1

For dependent aliases, the `userType` field of the `AliasRecord` contains a signature that indicates the direction of the reference. A backward alias has an alias `userType` of `'back'`. A forward alias has an alias `userType` of `'fore'`.

Custom data can be added to the dependent aliases to provide further identification. The `aliasSize` field contains the size of the alias record, and you can jump after the record to add your custom data.

QuickTime adds 8 bytes after the Alias Record. The first 8 bytes are reserved for QuickTime and describe the resource type and ID. For movies, the `ResType` is `'moov'`. The final two bytes are reserved. The custom data format is

```
struct    {
    ResType     refResource;        // set to 0 for non-QuickTime
    short        refResourceId;     // set to 0 for non-QuickTime
    short        reserved;        // Always set to 0
    };
```

You should not alter any custom alias that your application did not create. However, you can use the information to identify and delete dependent aliases.

Back to top

Working With Dependent Files

All the normal file rules apply to dependent files. Of course, you can delete them and copy them. But, in certain circumstances, you may need to operate on a referenced or referencing file's dependent aliases and Finder Information.

Deleting Dependent Files

Deletion of a dependent file means deleting all dependent aliases, clearing the shared bit of dependent files if necessary, and finally deleting the file itself. Deletion of a file by definition means deletion of one file, and one file alone. A dependent file deletion also includes the deletion of the dependent aliases and appropriately setting the shared bit. This three-step process can be quite complex. Here are the rules:

If QuickTime is installed, use `DeleteMovieFile` to delete movie files. `DeleteMovieFile` will perform the three-step process. QuickTime uses the File Manager to do its file handling, but, in addition, it adds the logic to handle dependent files. You can also use `DeleteMovieFile` on non-QuickTime files without any problems.

If QuickTime is not installed or you create your own dependent files, use the File and Alias Managers to delete dependent files. The use of the File Manager and Alias Manager to delete QuickTime movie files is strongly discouraged. But, lack of QuickTime and creating your own dependent files are two situations where you will have to perform the deletion yourself.

Since aliases are part of a dependent file, you can not work with dependent files on System 6 without QuickTime. QuickTime installs the Alias Manager on System 6 and `DeleteMovieFile` will then be able to work with movie files.

Removing Dependent Aliases

For this discussion, target file means the file to be deleted, and the alias file means the file at the end of the dependent alias. Also, this discussion covers the deletion of a target file with a forward dependency alias, because you use the same steps for a backward dependency alias. You just need to search for a backward dependency alias instead. The seven steps to delete a dependent file are as follows:

1. Open the target file.

For the target file you want to delete, you need to delete all dependent aliases. Thus, first you need to open the target file's resource fork. Be sure to have an FSSpec, because you will use it in step 4 below.

2. Search the target file's resource fork for forward dependent aliases.

For each alias in the target file, you need to see if it has a forward dependent alias by checking the alias' `UserType` field for `'fore'`. Then you need to retrieve the QuickTime custom data in order to be sure the target file is a movie file. If the custom data contains `'moov'` in the `ResType` field, then you know you have a QuickTime dependent file.

3. Search the alias file's resource fork for backward dependent aliases.

Resolve the forward dependency alias and open the resource fork. For each alias in the file, if it is a forward alias, get the custom data from it and compare it to the custom data you stored away. If it is a 'moov' resource and the resource ID matches, resolve the alias.

4. Compare `FSSpec` of the target file and the resolved backward dependent alias.

You now need to be sure you have the correct backward alias. You already have an `FSSpec` from the target file. You can create another `FSSpec` by resolving the backward alias. Call `FSSpecEqual` with both of these `FSSpecs`. If the `FSSpecs` are equal, you know you have the correct dependent alias.

5. Remove the backward dependent alias.

You can now remove the alias. Be sure to update your resources correctly and close the resource file of the alias file.

6. Check the shared bit and delete the file

As you perform these operations, be sure to keep track of other dependent aliases for the alias file. If your dependent alias was the only one, then the alias file loses its dependency and you should clear the shared bit.

7. Delete the target file.

Finally, you need to close the resource fork of the source file and delete it.

Copying Dependent Files

Copying a dependent file means creating a new file, and a new set of dependent aliases. In addition, the shared bit should be checked to be sure it is set.

Creating Dependent Files

Many current applications create dependent files, but they do not do it uniformly. Thus, it is recommended that you format dependent files like QuickTime dependent files. It is possible future versions of system software will further exploit this information. In addition, other applications will be able to understand your files if you support this format.

To create a dependent file, you need to create your files, create the dependent aliases, and set the shared bit. To create the dependent aliases, use the normal Alias Manager routines. Set the `userType` field of the alias to either `'fore'` or `'back'`. For the custom data for the dependent aliases, you need to put zero in the first 8 bytes after the alias' private data.

How About the Finder?

How about it? The GetInfo INIT that shipped with QuickTime 1.0 Developer's CD showed how Finder support for this feature could be supplied. If you dragged a dependent file to the trash, it would post a dialog that warned the user about the dependency (if the option key was held down, it did not). Also, it modified the Information dialog to display the dependent

files through use of pop-up menus. Without the GetInfo INIT, dependent files are *not* deleted using the method in this note. GetInfo works only with Systems 7.0.x, and does not work with system software version 7.1 or later.

Orphaned Dependent Files

Thus, the Finder without the INIT will not delete dependent files correctly. Fortunately, this inability is not terribly problematic. It means dependent aliases may be left around in files, and shared bits may be set incorrectly. These files are called *orphaned dependent files.* If your application works with dependent files, you may work with orphaned dependent files. Some dependent aliases will not be able to be resolved. Your application should be aware of this possibility and should be able to handle orphaned dependent files gracefully.

Cross-Platform Movie Files

QuickTime for Windows' movies are by definition single forked and self-contained. Therefore, dependent files do not apply to that platform.

Back to top

References

Inside Macintosh, Volume VI, Chapter 9, Finder Interface

Inside Macintosh, Volume VI, Chapter 27, Alias Manager

Inside Macintosh, QuickTime

Back to top

Downloadables

Acrobat version of this Note (K). Download
