Technical Note NW 12

AppleShare-able Applications and the Resource Manager

CONTENTS

Normally, applications on an AppleShare server volume cannot be executed by more than one user at a time. This technical note explains why, and tells how you can enable your application to be shared.

[Mar 01 1987]

The Resource Manager versus Shared Files

Part of the explanation of why applications are not automatically sharable is based on the design of the Resource Manager. The Resource Manager is a great little database. It was originally conceived as a way to keep applications localizable (a task it has performed admirably), and was found to be an excellent foundation for the Segment Loader, Font Manager, and a large part of the rest of the Macintosh operating system.

However, it was never designed to be a multi-user database. When the Resource Manager opens a resource file (such as an application), it reads the file's resource map into memory. This map remains in memory until the resource file is closed by the Segment Loader, which regains control when the application exits. Sometimes it is necessary to write the map out to disk; normally, this is only done by `UpdateResFile` and `CloseResFile`.

If two users opened the same resource file at the same time, and one of them had write access to the file and added a resource to it, the other user's Resource Manager wouldn't know about it; this would make the other user's copy of the file's original resource map invalid. This could cause (at least) a crash; if both users had write access, it's not unlikely that the resource file involved would become corrupted. Also, although you can tell the Resource Manager to write out an updated resource map, there's no way for another user to tell it to refresh the copy of the map in memory if the file changes.

What does all this have to do with running my application twice?

Your application is stored as a resource file; code segments, alert and dialog templates, etc., are resources. If you write to your application's resource file (for instance, to add configuration information, like print records), your application can't be shared.

In Apple's compatibility testing of existing applications (during development of AppleShare), we found quite a few applications, some of them quite popular, that wrote to their own resource files. So we decided, to improve the safety of using AppleShare, to always launch applications using a combination of access privileges such that only one user at a time could use a given application (these privileges will be discussed in a future Technical Note). In fact, AppleShare opens all resource files this way, unless the resource file is opened with `OpenRFPerm` and read-only permission is specified.

But my application doesn't write to itself!

We realize that many applications do not. However, there are other considerations (covered in detail, with suggestions for fixes, in "Application Development in a Shared Environment", available from APDA). In brief, here are the big ones we know about:

- Does your application create temporary files with fixed names in a fixed place (such as the directory containing

the application)? Without AppleShare's protection, two applications to use the same temporary file could be disastrous.
- Is your application at least "conscious" of the fact that it may be in a multi-user environment? For instance, does it work correctly if a volume containing an existing document is on a locked volume? Does it check all result codes returned from File Manager calls, and `ResError` after relevant Resource Manager calls?

OK, I follow the rules. What do I do to make my application sharable?

There is a flag in each file's Finder information (stored in the file's directory entry) known as the "shared" bit. If you set this bit on your application's resource file, the Finder will launch your application using read-only permissions; if anyone else launches your application, they'll also get it read-only (their Finder will see the same "shared" bit set.).

Three important warnings accompany this information:

- The definition of the "shared" bit was incorrect in previous releases of information and software from Apple. This includes the June 16, 1986 version of M.TB.Finder Flags (fixed in the March 2, 1987 version), as well as all versions of ResEdit before and including 1.1b3 (included with MPW 2.0). For now, the most reliable way to set this bit is to get the 1.1b3 version of ResEdit, use it to Get Info on your application, and check the box labeled "cached" (the incorrect documentation upon which ResEdit, et al.) was based called the real shared bit "cached"; the bit labeled as "shared" is the real cached bit [a currently unused but reserved bit which should be left clear]).
- By checking this bit, you're promising (to your users) that your application will work entirely correctly if launched by more than one user. This means that you follow the other rules, in addition to simply not writing to your application's own resource file. See "Application Development for a Shared Environment," and test carefully!
- Setting this bit has nothing to do with allowing your application's documents to be shared; you must design this feature into your application (it's not something that Apple system software can take care of behind your application's back). You should realize from reading this note, however, that if you store your document's data in resource files, you won't be able to allow multiple users to access them simultaneously.

Back to top

References

The Resource Manager, "Application Development in a Shared Environment"

M.TB.Finder Flags

Back to top

Downloadables

Acrobat version of this Note (K) Download

Back to top
