Apple II Technical Notes

Developer Technical Support

AppleTalk
#8: Using The @ Prefix

Written by: Jim Luther & Dan Strnad
September 1990

This Technical Note discusses use the @ prefix with multiple users.

Apple II computers on AppleShare networks feature a unique folder for each user on the server volume, called the user's folder. Server volumes containing these user's folders are called user volumes. User's folders exist on user volumes so that the system and applications have a standard place to store user-specific data on the network. All network volumes an Apple II can boot from are user volumes.

Under GS/OS, the @ prefix allows applications to automatically work with the user's folder. If a user launches your application from a server volume with a user volume mounted, GS/OS sets the @ prefix to the user's folder; otherwise it sets it to the application folder. The @ prefix can reduce design and coding effort for multilaunch features by providing the application with the system's best guess at where user-specific information should be stored. To safely use the user's folder feature, programmers need only remember to use the @ prefix with the GS/OS class 1 `Open` call (`requestAccess` = 1, 2, or 3). Using the @ prefix with the class 1 `Open` provides safe access to the file for as long as it remains open, without requiring any network-specific code.

Using the @ prefix is appropriate for applications that want to avoid

network-specific programming while being reasonably well-behaved in a network environment. For example, applications may store printer defaults in the @ directory or use it as a default when prompting the user to choose a directory.

There are situations writing data to a file in the @ directory could result in other users overwriting the data; however, applications may reasonably require that users not allow these situations to occur In Table 1, the third through fifth cases are all situations in which this problem could occur. For best results, when opening a file for writing with the @ prefix, use access privileges that deny write access to other users. The GS/OS class one `Open` call always does this when `requestAccess` is non-zero. Without this precaution of denying write access to other users, the user's data is not protected from being overwritten while it is in use.

Application launched from…	Is a user volume present?	User name	@ prefix set to…	Is this detectable?
local	maybe	any name	application prefix	yes
net	yes	(not guest)	user folder	yes
net	no	any name	application prefix	yes
net	yes	guest	guest folder	yes
net	yes	same as another user	user folder	special programming required

Table 1–Possible @ Prefix Configurations

Consider the third case. Although the application was launched from a server, the server does not contain a user's folder, nor is there any other mounted server containing a user's folder. In this case, if multiple users launch a single copy of the application from the same folder at the same time, each user's @ prefix would point to the same folder from which they all had launched the application. By denying other users write access when opening the file, you prevent users from overwriting each other's data. However, the other users are no longer prevented from overwriting the data when the user with write access closes the file, at which point a different user can write to the file. Therefore, using access privileges in combination with the @ prefix deters one user's data from being overwritten by another, but only so long as the file remains opened by the user with write access. This approach may provide sufficient protection for saving certain user configuration and preference information.

When saving work the user plans to resume later, this approach may not provide sufficient protection. In this situation, conflicts can also arise if the @ prefix is set to the application prefix rather than to the user's folder. It is up to the programmer to determine whether to use the @ prefix, how

to use it, and whether this level of protection is sufficient for the particular data involved.

In addition to using the @ prefix (or the user path to which it attempts to refer) with access that denies other users permission to write to the file, applications can check to see if the user path could point to the same folder for multiple users at the same time. To do this, the application first checks to see if it was launched across the network. This is the case when class one `GetFileInfo` on the user path returns `fileSysID` = $0D. If the application was launched across the network, the user path could be set the same for multiple users if the user has logged on as a guest (`UserInfo` returns a null `userName`, the fourth case in Table 1) or if you are using the @ prefix and the system has set it to the application prefix on a non-user network volume (error $60 from `GetUserPath`, the third case in Table 1). If the application determines that the user prefix may be set the same for multiple users, then it could use an alternate approach to determine where the data is to be stored, by prompting the user for example.

Although it would be comparatively difficult for an application to determine whether multiple users with the same name were running the application from the same server (the fifth case in Table 1), the documentation for the application could warn against this. The system does not provide any specific information about when this condition occurs.

One More Caution

One other caution to observe when using the @ prefix: since other applications are also storing data in the same user's folder, each application should be careful to reference distinct files. Regardless of how the application chooses to do this—by checking that the file being created does not already exist, by choosing a distinct name for the file, or by some other method—it should usually operate only on files of its own creation.

Programmers should keep in mind that the @ prefix is provided as a programming convenience. The AppleShare FST also provides the `GetUserPath` and `UserInfo` calls. In combination with `GetFileInfo`, these calls allow programmers to devise other, more customized approaches for determining where to save the user's data.

Further Reference

- *AppleShare Programmer's Guide for the Apple II Family*
- *GS/OS Reference*
- GS/OS Technical Note #10, How Applications Find Their Files