Apple II Technical Notes

Developer Technical Support

GS/OS
#4: A GS/OS State of Mind

Revised by: Matt Deatherage March 1991
Written by: Matt Deatherage January 1989

This Technical Note discusses GS/OS concepts and practices.
Changes singe July 1989: Includes more information about thinking for non-ProDOS file systems.

Although GS/OS bears many similarities to ProDOS, GS/OS is a much wider-reaching operating system, working not only with multiple file systems but also with character devices. Some things which work under ProDOS cause problems under GS/OS, and application programmers need to be aware of the differences, particularly those developing text-based programs.

GS/OS Hints

Be aware of character devices. A legal GS/OS pathname, perhaps entered by a user in response to a prompt, could map to a character device, with potentially disastrous results. Error $58, `Not a Block Device`, can protect you against this on many calls, including `Create`, but you must still take precaution. `DInfo` tells you if a device is a character device or block device; bit seven of the `characteristics` word is set if the device is a block device.

Don't preprocess pathnames. A user input routine which prevents users from entering pathnames that don't follow ProDOS syntax may help prevent `Illegal Pathname`

`Syntax` errors, but it also keeps users from creating files on non-ProDOS disks with anything but ProDOS pathname syntax, and it could keep them from accessing files on non-ProDOS disks which they created with another GS/OS application. Since the only FST which allowed you to write to a device under System Software 4.0 was ProDOS, you didn't see this problem right away. However, System Software 5.0 includes an AppleShare FST which, compared to ProDOS, is fast and loose with pathnames. "How about an anti-ProDOS name?" is a legal AppleShare filename. To allow compatibility with present and future non-ProDOS FSTs, Apple suggests you pass user-entered pathnames directly to GS/OS, with no application preprocessing.

Remember that under GS/OS both colons and slashes are valid separators, and colons can only be separators. In addition, all eight bits of each byte of a pathname are significant. Refer to *GS/OS Reference*, Volume 1 for more information on GS/OS pathname syntax. Using all eight bits of each byte may be particularly difficult for text-based applications, which have no way to force the standard Apple II character set to display characters such as sigma (∑) or the copyright symbol (©); they can fiddle to get characters like the sterling pound sign (£) and an Apple (□). Some programs may wish to adopt special typographical conventions for these special characters while others may choose not to create files with such characters in their names. These programs could present the user with a list of existing filenames (with some substitution for the characters which are unavailable), while providing a method of choosing one, to retrieve such files. Any way around this problem for a text-based program will be less than optimal.

Avoid the Text Tools and all slot dependencies. Preliminary GS/OS documentation points to a System Service call named `DYN_SLOT_ARBITER`. This mechanism, which is not fully implemented in System Software 5.0, eventually will allow the operating system to use internal ports and external slots for the same "slot" in the same session, instead of requiring the user to reboot the system to safely change between ports and slots. Applications which have hard-coded slot dependencies (as the Text Tools unfortunately require) make this transition very difficult, both for GS/OS and for the applications and users. We recommend that applications use the GS/OS loaded and generated character device drivers for text output. A `DInfo` call will tell you what slot or port a driver controls, and whether or not it is a character device.

Avoid other file system dependencies. Many of the things ProDOS programmers are used to as facts of life just are not true any longer. For example, filenames don't have to be 15 characters or less under GS/OS. When making class one calls, GS/OS will tell you if you don't have enough room for the pathname by returning a `Buffer Too Small` error ($4F). Avoiding file system dependencies means handling this error intelligently: if you receive it, allocate more space for the buffer and try the call again. GS/OS will tell you how much space is needed. If you absolutely must hard code pathnames, such as volume names, be sure to use the colon as the separator, because if you do not, filenames with slashes will cause problems. Similarly, don't assume any of the following:

- There can only be 51 files in the volume directory

- All devices are named "`.Dn,`" where `n` is the device number
- All blocks are 512 bytes long
- All devices are block devices
- Any other ProDOS-specific characteristics

Your application may have hidden file system assumptions as well. For example, while a directory behaves like a directory under all GS/OS file system translators, reading from a directory is not always as fast as it is for ProDOS disks. ProDOS directories are fairly linear and can be searched quickly; but other file systems may have more complicated directory structures (HFS and AppleShare, for example, have B-trees that store directory entries in alphabetical order). To get optimal speed, try to do as many `GetDirEntry` calls as you can in succession without other GS/OS calls intervening—this allows Apple to optimize file system translators for fast directory reading.

Also remember that other file systems may not support the concept of orderable directories, so don't depend on directory order in your application.

Don't hog all of the memory. While this is never a good idea on the IIGS, it's even worse under GS/OS. To process things like pathnames, GS/OS allocates memory through the Memory Manager. If you've allocated all of available memory (i.e., for a disk copy procedure), GS/OS will be forced to return an `Out of Memory` error ($54). If the condition is so severe that GS/OS can no longer function, it will return a fatal GS/OS error with an ID = 2, and the user will be asked to restart the system.

(A common cause of fatal GS/OS error 2 during development is using a length byte instead of a length word on a class one string. Doing so almost always causes the first word to be greater than 8K, which is the maximum length of pathnames under GS/OS. GS/OS then dies for your enjoyment, as it is unable to allocate the memory for the pathname because it's too big, even if more than 8K is available.)

Hard code as little as possible. Even seemingly static things like device names should not be hard coded, since a new loaded driver could change the name of the same device at any time. Also, it may be possible in the future for users to rename devices.

Only ask for the access you need. If you're just going to read a file, make a call to `Open` the file with read permission only. In file systems where access privileges mean more than they traditionally have in ProDOS (where things are usually "Locked" or "Unlocked"), this could save some trouble. For example, AppleShare allows the same file to be opened multiple times as long as each open is with read-only access. If your program is only going to read a file, opening it with read and write access needlessly denies others on the server access to the file.

Copy all GS/OS information with files. Applications that copy files need to do more than

copy the data fork of the file. If the file is extended, the resource fork of the file should be copied as well. In addition, when requested, each FST returns an `option_list` that contains information specific to the host file system that GS/OS does not use (i.e., AppleShare's `option_list` includes Finder information and access privileges). Calls to `GetFileInfo` and `Open` can return the `option_list`, while a call to `SetFileInfo` can set it. An FST will not set parameters in the `option_list` which should not be altered (just as `SetFileInfo` skips the EOF fields in `GetFileInfo` records). To ensure that the duplicate has as much host file system information from the original as can reasonably be transferred, always copy the `option_list`.

However, if you **want** to change something in an existing file's `GetFileInfo` list, do **not** use an `option_list`. The `option_list` could override the other parameters to `SetFileInfo` without your knowledge.

Further Reference

- *GS/OS Reference*, Volumes 1 and 2