Do You Copy?

Are you copying all the information about your files that you need?

The Macintosh environment has always been feature rich, and that extends to its files and the attributes and file metadata the filesystem maintains for them. OS X has introduced Macintosh to a wider array of application environments which aside from that of the classical include beasts such as Carbon, Cocoa, BSD, X-Windows, Web Objects, Java, QuickTime, and now Rossetta. And OS X supports different filesystems that support different metadata. All of these may see and consider files differently than others. This results in files being copied rather differently.

This article provides is a short attempt to attune application developers, scripters, systems programmers and filesystem architects to considerations relating to file metadata stored by the filesystems on Mac OS X.

Philosophic Questions
In many respects it is a philosophic question as to what each environment on the Mac should properly copy. Some environments may not normally or naturally have access to all file metadata that OS X maintains for any given file. Named Extended Attributes may not be accessible through Carbon, Cocoa and Core Foundations while they may be accessible from BSD. The nature of the operation may color and determine what metadata is important and what makes sense for preserving as a copy or whether the metadata should be reflective of a whole new file.

Consider the subtle differences of what metadata, or even files, get maintained as part of the different functions of operations like copies vs archives vs backups vs clones vs sync vs snapshots, et cetera. Is a copied file a new file or is the copy merely a replica or clone? This isn't as easy of a question as it first seems. For instance if you copy a file from one location to another does it make sense to maintain any ACLs it may have had or shouldn't the new file have it's own, new ACL. But now what if that "copy" was a backup or clone and part of larger set of files in its directory? Wouldn't you then expect the ACLs to be maintained? Likewise if you're backing up a *set* of files would you expect metadata associated with the directory (like sort order, icon positions, etc) to be maintained for the files? Yet these are stored in places like .DS_Store or the Desktop Database not as metadata for the file itself.

What about when we copy files to non-HFS+ filesystems? What should get maintained? What should be expected that if copied off, gets copied back? The Macintosh has defined the Apple Double format to store a file non-natively. Is it right for a Windows sysadmin to delete ._file's or .DS_Store's found on their filesystems; is this "clutter"? To the Mac user who's lost the CODE resource of an application file, icons or font resources, a file's creation date, or creator/type you may end up with torches and pitchforks at your door if you're inconsiderate, and confusing if your tools are inconsistent. But should an ACL get copied to a foreign filesystem? Maybe not, but other Extended Attributes clearly are in Apple Doubles.

Moreover consider what file creation, copying, and modification may mean. For instance consider that cat > file is often seen as creating a new file, but that if ‘file’ already existed then this operation is in fact a “modification”.

Lastly, in the future we may be limited as to *what* we may and may not copy through DRM. This may become a technical rather than semantic issue as “secure” (as in secure for to the owners of the content) filesystems and files are deployed.

Mac OS X File Metadata and Attributes

Briefly let’s run down some of the most significant file metadata and attributes common to Mac OS X.

Data Streams

The data fork is well known, but all Macintosh files have at least two forks, a data fork and a resource fork, either of which may be of zero length. Implied is that both exist. Under HFS+ multiple named streams may exist, which may be named, but the data and resource streams may not be renamed. Until Tiger this was merely semantics.

Traditional views of files are just a single byte stream of named data (like found on a tape) with BOFs and EOFs and little to no metadata. The metadata was just “accounting” information used by the filesystem, it wasn’t really “the data”. But over the years additional information was tacked on by filesystems. User’s got used to this additional metadata for cataloging and sysadmins got used to preserving them on backups. Environments became more complex and filesytems started offering better ways to store ancillary data. The Macintosh isn’t unique in offering multiple streams per file. Even Windows’ NTFS has such provisions. HFS+ defined an architecture for “unlimited” numbers of streams or forks using the Attributes B-tree structure if the volume.

Under OS X the mandatory data and resource forks of a Macintosh file are exposed on HFS+ volumes as filename and filename/..namedfork/rsrc to BSD environments and as filename and ._filename when “split” (as for use on foreign filesystems.) It also introduced tools for splitting and recombining forks split into what are arguably Apple Doubles.

Mac OS X 10.4 Tiger introduced the concept of Extended Attributes which implemented and exposed the additional named forks or streams on HFS+ volumes and through the ‘split’ fork ._filename on foreign filesystems. ACLs are in turn implemented as a privately named and protected Extended Attribute and hence are a third fork that files may have on filesystems that have ACLs enabled. Additionally any number of additional “Extended Attributes” (forks) may now be added to or read from a file with setxattr(), getxattr(), listxattr() and removexattr().

Finder Flags

See `man GetFileInfo` for details, but these include file creator and type, attributes bits (such as locked, stationary, invisible), creation date and modification date. Stored in ._filename Apple Double half on filesystems other than HFS and its derivatives.

POSIX
Files have ownerships (user and group) and permissions. Symbolic links also have these attributes. Unix files also traditionally carry dates such as atime, mtime and ctime, namely the time of last access, last time file modified, and change time (last time the file's inode was changed), respectively. Note that ctime *changes* after operations like chmod.

Creation Dates
The subject of creation dates on the Mac warrants detailed discussion in itself. The Mac has always maintained file "creation date" metadata, it's a Finder file attribute, copied along with the file by the finder, displayed in the Finder's Get Info dialog and important enough that it is supposed to be included as part of the Apple Double file information stored in a ._file. But while Mac users are long familiar with it and often use it to categorize files it's a rather foreign concept to unix (mostly because it traditionally never existed there) and some unix pundits believe it's just woolly-thinking (see http://mail-index.netbsd.org/netbsd-users/2000/11/22/0000.html) or at best the product of the confused (see http://toadstool.se/journal/2006/01/11/the-fallacy-of-ctime). Other unices and filesystems support a btime (birth time) with varying degrees of success.

However, Mac users are used to creation dates, and may get rather agitated if they're lost or, say replaced with the modification time (which seems to be the OS X BSD environment behavior.) As such, applications should strongly consider properly handling this metadata or if they do mangle it do so in a manner that lends itself to notice (like beginning of epoch.)

For a more detailed commentary of Mac file creation dates see
http://blog.plasticsfuture.org/2006/06/27/mac-os-file-creation-dates/

BSD Flags
BSD traditionally has maintained a series of flags associated with properties of a file as manipulated by `chflags`. These include flags such as the 'system immutable flag' (schg) that prohibit a file from being modified, or the 'archive flag' (arch).

The classical Desktop Database and OS X .DS_Store files
Traditionally the Macintosh stored certain metadata regarding files in the Desktop Database. Under Mac OS X the .DS_Store file associated with a directory contains similar information such as directory background info, the position or order of files and their icons within a directory, an more. When copying a directory you should probably consider copying these.

Lastly there's one Finder level piece of information store disassociated from the file, which brings us to...

Spotlight Metadata
Tiger introduced us to Spotlight and a huge amount of application specific metadata stored associated with a file based on its content. This data is indexed by the kernel through 'importers' as a part of normal operations on the file.

While most of this doesn't need preserved since it is indexed as needed by the kernel, one attribute exposed to the Finder and stored in .DS_Store is what traditionally was known as the file Comments, called under Tiger the Spotlight Comments and is displayed through the Finder's Get Info dialog.

How to preserve file metadata

There a few primary methods by which file metadata get preserved:

- copied natively
- preserved in special containers or lists
- preserved by images of the volumes that support them (e.g. .dmg's)
- split as Apple Doubles
- BSD copies files using copyfile() which splits to Apple Doubles as necessary either internally or externally (with varying results)

Note that currently copyfile() munges the modified date into the creation date field of its Apple Doubles.

What needs copied? What doesn't?

While mostly a philosophic issue, it's probably safe to say that traditional Macintosh attributes such as creator and type are becoming less important since OS X has various mechanisms for associating files with their applications (sometimes at the cost of additional metadata) while other metadata like creation date should arguably be maintained. Spotlight metadata looks like a good candidate for general exclusion since the kernel will automagically recreate this for installed application importers on the target machine, but all Mac OS X systems might not have the same importers installed. This may or may not matter. Old Desktop Database files probably aren't an issue either as Classic withers or is mooted by Intel Macs, but .DS_Stores should probably be maintained if you're backing up a volume. Mandatory file forks should always be copied but while these are Extended Attributes you might consider not copying the other forks that may exist, such as ACLs, depending on if you're backing up or cloning or if you're just duplicating the file elsewhere on a volume. It's hard to tell currently if other Extended Attributes need copied since they're rarely seen in action.

What Copies What?

Let's take a look at what metadata some common tools and their operations preserve. The following is not designed to be a "report card" and it's not my goal to rank or rate a tool for its handling of the data, but sysadmins should be aware of how operations they perform may affect their data.

Coverage of various thrid-party backup applications is beyond the scope of this article, but it seems they fare rather poorly with regard to not preserving Extended Attributes aside from resource forks. Creation dates are a mixed bag and may vary within the application based on operation (sometimes intentionally.) For additional discussion see http://blog.plasticsfutures.org/2006/03/05/the-state-of-backup-and-cloning-tools-under-mac-os-x/

In general copyfile(), which most all BSD tools rely on, does not perform consistently compared to the operation of the Finder. Creation date is clobbered by the modification date despite that copyfile() attempts to use Apple Doubles 'internally' and externally to foreign file systems and the Apple Double format specifically calls for the creation date. The various tools that use copyfile() then quite often fail to properly maintain internal or synthetic files resulting in lost Extended Attributes and other data. In rare cases the tool may crash, as in the instance of trying to `rsync -aE` a file with both ACLs and additional non-resource fork, named Extended Attributes.

Apple System Restore (asr) in device mode manages to copy all the tested data, but this is to be expected since it is essentially a complete copy of the device. In file mode however asr, Disk Utility, and hdiutil have digressed in their behavior with OS X 10.4.6. Where previously it maintained locks (the Finder "L" attribute and BSD uchg flags), Extended Attributes and ACLs it no longer preserves them, though this may change in a future update or (although this makes packages and dmgs problematic) it may be the intended behavior.

`dd` fares pretty much as you'd expect. At least philosophically it's understandable.

Who Copies What

	Finder Flags			Dates		Extended Attributes			BSD Flags	POSIX			File ID	Finder
	attribute bits	creator/type	locked	creation	modification	rsrc fork	ACLs	xattr		owners	permissions	symlink owners	(for aliases)	Comments
Finder	X	X	X	X	X	X	8	X	3	0	X	0	0	X
cp -Rp	X	X	X	1	X	X	0	X	X	X	X	0	0	X
ditto	X	X	X	1	X	X	0	X	0	X	X	0	0	X
rsync -aE	0	0	0	1	X	0	2	0	0	X	X	X	0	X
tar	0	0	0	1	X	4	2	4	0	X	X	X	0	X
asr (device mode)	X	X	X	X	X	X	X	X	X	X	X	X	X	X
asr (file mode)	X	X	0	1	X	X	0	0	0	X	X	0	0	X
Disk Utility dmg	X	X	0	1	X	X	0	0	0	9	X	0	0	X
hdiutil dmg	X	X	0	1	X	X	0	0	0	9	X	0	0	X
dd (6)	7	7	7	7	7	7	7	7	7	7	7	N/A	N/A	5

NOTES:
Darwin Kernel Version 8.7.0: Fri May 26 15:20:53 PDT 2006; root:xnu-792.6.76.obj~1/RELEASE_PPC

Finder Comments only copied if whole directory copied (that is not if you just copy the file, you must copy the .DS_Store)
1 - replaced w modification date
2 - xattr info munged, doesn't display "+" in ls but ls -le displays ACL, attribute becomes non-private
3 - uchg flags set from Finder locked ("L" attribute) preserved, others BSD flags appear lost
4 - generates warning, can't stat file, not preserved
5 - even if we dd .DS_Store file
6 - dd ONLY copies data
7 - dd writes whole new files
8 - no longer displays "+" in ls -le but ls -le displays ACL
9 - requires "-owners on"

Dan Shoop is principal researcher at iWiring (www.iwiring.net) and a Systems and Networks Architect for US Technical Services (ustsvs.com), both of which provide service and support for the Macintosh and other computer systems. He may be reached at shoop@iwiring.net.