

*Demystifying the Geekier Side of Mac OS X*

# Mac OS X Tiger

*for*  
Unix  
Geeks



**O'REILLY®**

*Brian Jepson & Ernest E. Rothman*

---

# Searching and Metadata

If a Unix Geek needs to find something, she'll probably use *locate* or *find*, depending on what she's looking for. Because *locate* is based on a static database that's only regenerated periodically (see "Scheduling Tasks in Chapter 4), it would be the choice for things that don't change a lot (virtually anything in */usr*). It's also much faster because it has that database to consult. And trusty old *find*, slow as molasses, is what you need when you need more control over the search or when you're looking for something that *locate* doesn't know about, such as files that have been created recently.

But Tiger introduces a new search capability, Spotlight, which stores and sifts through file metadata faster than a herd of sheep can clear a field. Spotlight comes in two forms: a GUI interface accessible from the menubar, and a suite of command-line utilities. This chapter introduces you to Spotlight and shows you how to take advantage of all it has to offer.

## Spotlight

Remember the relentless disk grinding you heard after you first installed the operating system? That was Spotlight creating its initial database. Spotlight is a repository of metadata for certain types of files—Spotlight gathers information about any file (or data record, such as an iCal event) for which it has an *importer* (an operating system plug-in that extracts metadata from a document). To see all the importers on your system, look in */System/Library/Spotlight* and */Library/Spotlight*.

By default, Spotlight has importers for the following files and data:

- Address Book records
- AppleWorks files
- Applications

- Audio files
- Safari bookmarks
- iChat transcripts
- Fonts
- iCal events
- Images
- Keynote presentations
- Mail messages
- Microsoft Office documents
- Pages documents
- PDF and PostScript files
- QuickTime movies
- RTF documents
- Source code
- System preferences

To perform a spotlight query, simply click the magnifying glass icon in the upper right of the menu bar or press **⌘-Space**. A Spotlight search field drops down, in which you enter a search term, as shown in Figure 2-1.

You can get a more detailed Spotlight search window by pressing **Option-⌘-Space**. This window, shown in Figure 2-2, lets you configure a number of aspects of your search, such as location, date, and result grouping.

## Performing Spotlight Searches

Unix geeks might never use Spotlight if Mac OS X didn't include some command-line goodies for performing searches. You can perform a simple Spotlight search from the shell with the following syntax:

```
mdfind term
```

For example:

```
$ mdfind burroughs
/Developer/Documentation/DeveloperTools/Tcl/Trf/bz2.html
/Volumes/Macintosh HD/Users/bjepson/Music/iTunes/iTunes Music/William
  S. Burroughs
/Volumes/Macintosh HD/Users/bjepson/Music/iTunes/iTunes Music/William
  S. Burroughs/Dead City Radio/02 A Thanksgiving Prayer.m4p
/Volumes/Macintosh HD/Users/bjepson/Music/iTunes/iTunes Music Library.xml
/Volumes/Macintosh HD/Users/bjepson/Sites/radio/2003/04/11.html
```

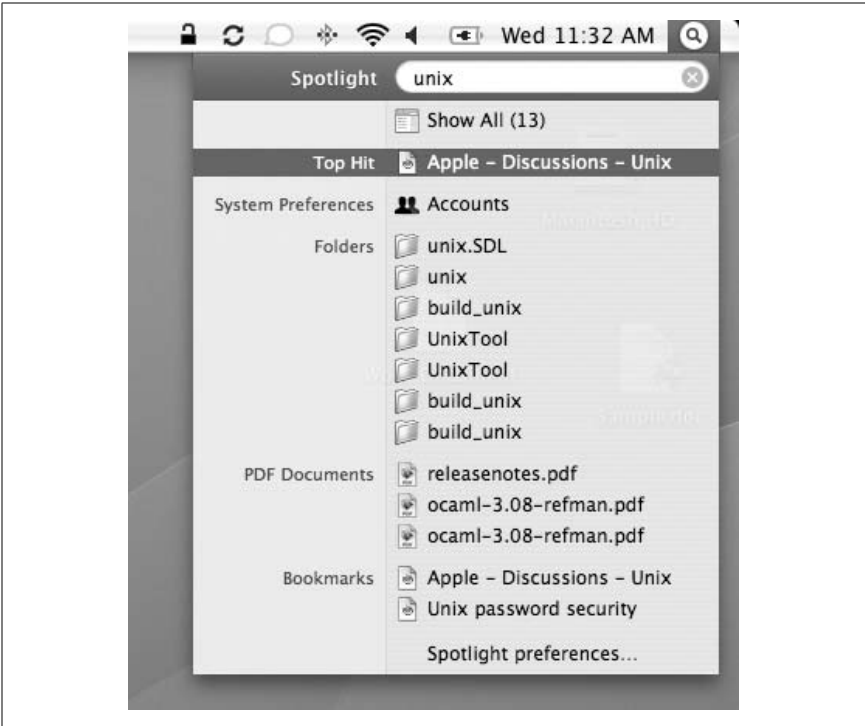


Figure 2-1. Using the Spotlight menu

If you have a good idea of where you want to search, you can use the *-onlyin* option as shown here:

```
$ mdfind -onlyin /Developer burroughs
/Developer/Documentation/DeveloperTools/Tcl/Trf/bz2.html
```

You can use the *-live* option to update the results in real time as they change, and as quickly as Spotlight can index them.

Although you can find interesting results with simple keyword searches, you can refine your search by specifying any of the metadata attribute keys. For example, to find all the songs written by Robert Hunter, you could use this search:

```
$ mdfind "kMDItemComposer == '*Robert Hunter*'"
/Users/bjepson/Music/iTunes/iTunes Music/Grateful Dead/Hundred Year
Hall (Disc 1) [Live]/1-01 Bertha.m4a
/Users/bjepson/Music/iTunes/iTunes Music/Grateful Dead/Hundred Year
Hall (Disc 1) [Live]/1-04 China Cat Sunflower.m4a
/Users/bjepson/Music/iTunes/iTunes Music/Grateful Dead/Hundred Year
Hall (Disc 1) [Live]/1-06 Jack Straw.m4a
/Users/bjepson/Music/iTunes/iTunes Music/Grateful Dead/Hundred Year
Hall (Disc 1) [Live]/1-08 Playing In The Band.m4a
```

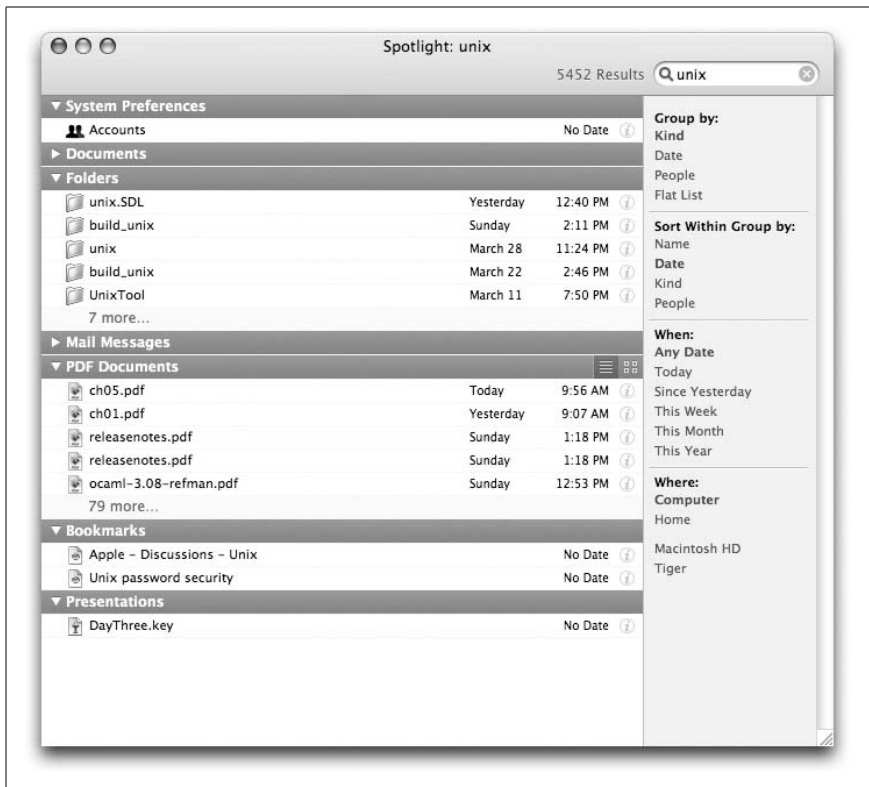


Figure 2-2. Searching with the Spotlight window

```

/Users/bjepson/Music/iTunes/iTunes Music/Grateful Dead/Dick's Picks
  Volume 12 (Disc 1)/1-02 China Cat Sunflower _.m4a
/Users/bjepson/Music/iTunes/iTunes Music/Grateful Dead/Dick's Picks
  Volume 12 (Disc 1)/1-06 Truckin' _.m4a
[... and so forth ...]

```

Without the wildcard characters (\*), you wouldn't match a thing, since Robert Hunter coauthored all those songs with Jerry Garcia. You can perform more complex queries with *mdfind*, as well. For example, the following query uses the and (&&) operator to combine two search criteria: the composer name contains "Jerry" and the performer (author) name does not contain "Grateful Dead":

```

$ mdfind "kMDItemComposer == '*Jerry*' && \
  kMDItemAuthors != '*Grateful Dead'"
/Users/bjepson/Music/iTunes/iTunes Music/Compilations/Box of Pearls - The
  Janis Joplin Collection/3-11 Piece of My Heart (Live at Woodstock).m4p
/Users/bjepson/Music/iTunes/iTunes Music/The Who/Live At Leeds/11
  Summertime Blues.m4a

```

## Inspecting a File's Attributes

Now that we've found a couple songs written by someone named Jerry who's not in the Grateful Dead, how do we figure out what his deal is? The *mdls* utility lets you see all of the metadata for a given file:

```
$ cd ~/Music/iTunes/iTunes\ Music/Compilations/
$ cd Box\ of\ Pearls\ -\ The\ Janis\ Joplin\ Collection/
$ mdls 3-11\ Piece\ of\ My\ Heart\ \ (Live\ at\ Woodstock\).m4p
3-11 Piece of My Heart (Live at Woodstock).m4p -----
kMDItemAlbum                = "Box of Pearls - The Janis Joplin
Collection"
kMDItemAttributeChangeDate   = 2005-02-19 17:43:08 -0500
kMDItemAudioBitRate          = 125240
kMDItemAudioChannelCount     = 2
kMDItemAudioTrackNumber      = 11
kMDItemAuthors               = ("Janis Joplin")
kMDItemCodecs                = ("")
kMDItemComposer              = "Bert Berns & Jerry Ragovoy"
kMDItemContentCreationDate   = 2004-04-02 11:46:40 -0500
kMDItemContentModificationDate = 2004-04-02 11:46:40 -0500
kMDItemContentType           = "com.apple.protected-mpeg-4-audio"
kMDItemContentTypeTree       = (
    "com.apple.protected-mpeg-4-audio",
    "public.audio",
    "public.audiovisual-content",
    "public.data",
    "public.item",
    "public.content"
)
kMDItemDisplayName           = "3-11 Piece of My Heart (Live at
Woodstock).m4p"
kMDItemDurationSeconds       = 391.3483333333334
kMDItemFSCContentChangeDate  = 2004-04-02 11:46:40 -0500
kMDItemFSCreationDate        = 2004-04-02 11:46:40 -0500
kMDItemFSCreatorCode         = 1752133483
kMDItemFSFinderFlags         = 0
kMDItemFSInvisible           = 0
kMDItemFSLabel               = 0
kMDItemFSName                = "3-11 Piece of My Heart (Live at
Woodstock).m4p"
kMDItemFSNodeCount           = 0
kMDItemFSOwnerGroupID        = 501
kMDItemFSOwnerUserID         = 501
kMDItemFSSize                 = 6522800
kMDItemFSTypeCode            = 1295274016
kMDItemID                    = 128067
kMDItemKind                   = "MPEG-4 Audio File (Protected)"
kMDItemLastUsedDate          = 2004-04-02 11:46:40 -0500
kMDItemMediaTypes            = (Sound)
kMDItemMusicalGenre          = "Rock"
```

```

kMDItemStreamable           = 0
kMDItemTitle                 = "Piece of My Heart (Live at Woodstock)"
kMDItemTotalBitRate         = 125240
kMDItemUsedDates             = (2004-04-02 11:46:40 -0500)

```

That's a lot of information, but this sampling gives you an idea of what sort of search terms you can use with your *mdfind* queries. Table 2-1 lists some of the most common metadata attributes, but as you can see from the music file example, importers (in this case, the iTunes importer) are free to define their own attributes.

Keep in mind an important distinction when speaking of metadata: the owner (in terms of file system permissions) of the file is not necessarily its author. For example, if you rip an MP3 file from a CD-ROM, you're the owner. However, iTunes consults Cddb (the Gracenote CD Database, located at [http://www.gracenote.com/gn\\_products/cddb](http://www.gracenote.com/gn_products/cddb)) and uses the information it finds there to determine the authors of the file. On the other hand, if you create a Word document on your Mac, you'll not only be the owner of the file, but you're also the author. Another way to think about this is that Spotlight metadata is not so much about *files*, but the *contents* of files.

Table 2-1. Common Spotlight metadata attributes

Attribute	Description
kMDItemAttributeChangeDate	The date and time that a metadata attribute was last changed.
kMDItemAudiences	The intended audience of the file.
kMDItemAuthors	The authors of the document.
kMDItemCity	The document's city of origin.
kMDItemComment	Comments regarding the document.
kMDItemContactKeywords	A list of contacts associated with the document.
kMDItemContentCreationDate	The document's creation date.
kMDItemContentModificationDate	Last modification date of the document.
kMDItemContentType	The qualified content type of the document, such as <i>com.adobe.pdf</i> for PDF files and <i>com.apple.protected-mpeg-4-audio</i> for an Apple Advanced Audio Coding (AAC) file.
kMDItemContributors	Contributors to this document.
kMDItemCopyright	The copyright owner.
kMDItemCountry	The document's country of origin.
kMDItemCoverage	The scope of the document, such as a geographical location or a period of time.
kMDItemCreator	The application that created the document.
kMDItemDescription	A description of the document.
kMDItemDueDate	Due date for the item represented by the document.

Table 2-1. Common Spotlight metadata attributes (continued)

Attribute	Description
kMDItemDurationSeconds	Duration (in seconds) of the document.
kMDItemEmailAddresses	Email addresses associated with this document.
kMDItemEncodingApplications	The name of the application (such as “Acrobat Distiller”) that was responsible for converting the document in its current form.
kMDItemFinderComment	This contains any Finder comments for the document.
kMDItemFonts	Fonts used in the document.
kMDItemHeadline	A headline-style synopsis of the document.
kMDItemInstantMessageAddresses	IM addresses/screen names associated with the document.
kMDItemInstructions	Special instructions or warnings associated with this document.
kMDItemKeywords	Keywords associated with the document.
kMDItemKind	Describes the kind of document, such as “iCal Event.”
kMDItemLanguages	Language of the document.
kMDItemLastUsedDate	The date and time the document was last opened.
kMDItemNumberOfPages	Page count of this document.
kMDItemOrganizations	The organization that created the document.
kMDItemPageHeight	Height of the document’s page layout in points.
kMDItemPageWidth	Width of the document’s page layout in points.
kMDItemPhoneNumbers	Phone numbers associated with the document.
kMDItemProjects	Names of projects (other documents such as an iMovie project) that this document is associated with.
kMDItemPublishers	The publisher of the document.
kMDItemRecipients	The recipient of the document.
kMDItemRights	A link to the statement of rights (such as a Creative Commons or old-school copyright license) that govern the use of the document.
kMDItemSecurityMethod	Encryption method used on the document.
kMDItemStarRating	Rating of the document (as in the iTunes “star” rating).
kMDItemStateOrProvince	The document’s state or province of origin.
kMDItemTitle	The title.
kMDItemVersion	The version number.
kMDItemWhereFroms	Where the document came from, such as a URI or email address.

## Managing Spotlight

Spotlight is modestly configurable; you can use System Preferences → Spotlight to control the order in which results are presented, exclude certain file

types, and specify directories that must never be indexed. You can do quite a bit from the shell prompt as well.

The *mdutil* command controls Spotlight settings on a volume-by-volume basis, and *mdimport* lets you work with the various importers installed on your system. For example, *mdutil* can turn indexing on or off for an entire volume with the *-i* option (it takes an argument of *on* or *off*):

```
# mdutil -i off /Volumes/Macintosh\ HD
/Volumes/Macintosh HD:
    Indexing disabled for volume.
```

This setting is persistent across reboots. You can inspect a volume's setting with the *-s* option:

```
# mdutil -s /Volumes/Macintosh\ HD/
/Volumes/Macintosh HD/:
    Status: Indexing Disabled
```

You can use *mdimport* to list all the importers installed on your system:

```
$ mdimport -L
2005-02-26 22:10:53.296 mdimport[268] Paths: id(501) (
    "/System/Library/Spotlight/Image.mdimporter",
    "/System/Library/Spotlight/Audio.mdimporter",
    "/System/Library/Spotlight/Font.mdimporter",
    "/System/Library/Spotlight/PS.mdimporter",
    "/Library/Spotlight/Microsoft Office.mdimporter",
    [... and so forth ...]
```

And if the list of attributes in Table 2-1 isn't enough to keep you busy, you can also use *mdimport* to list all the attributes supported by the importers on your system:

```
$ mdimport -A
'kMDItemAcquisitionMake'          'Device make'
    'Make of the device used to acquire this document'
'kMDItemAcquisitionModel'        'Device model'
    'Model of the device was used to acquire this document'
'kMDItemAlbum'                   'Album'
    'Title for a collection of media, such as a record album'
'kMDItemAperture'                'Aperture'
    'Aperture setting of the camera when the picture was taken'
[... and so forth ...]
```

*mdimport* also has a number of features of interest to people developing their own metadata importers. For example:

- *-X* prints out an XML schema for the metadata on your system
- *-f* forces *mdimport* to import a directory or file (overriding any exceptions you've made in System Preferences)
- *-p* displays performance statistics for a run of *mdimport*

## Resource Forks and HFS+ Metadata

Apple's HFS+ filesystem has been stashing metadata away since its introduction in Mac OS X 8.1. *Resource forks* are generally invisible portions of files used for stashing additional information (the primary portion of the file—indeed the only part of a file most Unix Geeks are used to thinking about—is called the *data fork*). Before Mac OS X, resource forks contained application resources. These are now contained in the application bundle itself, although resource forks are still used in classic applications and a few odd places (such as text clippings, which you can create by dragging and dropping text selections to the Finder).

You can inspect a resource fork by appending */rsrc* to a file name, as in:

```
$ ls -l Sample.textClipping
-rw-r--r--  1 bjepson  bjepson  0 Feb 27 11:05 Sample.textClipping
$ ls -l Sample.textClipping/rsrc
-rw-r--r--  1 bjepson  bjepson 1770 Feb 27 11:05 Sample.textClipping/rsrc
```

The contents of a resource fork, even for something simple like a text clipping, are not necessarily human-readable, but there's usually something you can dig out:

```
$ file Sample.textClipping/rsrc
Sample.textClipping/rsrc: ms-windows icon resource
$ strings Sample.textClipping/rsrc
KApple's HFS+ filesystem has been stashing metadata away since its
introduction in Mac OS X 8.1. Resource forks are generally invisible
portionsof files used for stashing additional information (the primary
portion of the...
```

Mac OS X Tiger also makes use of HFS+ metadata, which consists of extended attributes that are associated with files. For example, if you look at the root of your Mac's hard drive in the Finder, you'll only see a small subset of the files (at the very least, Library, System, Applications, Users). But if you drop down into the Terminal, there are plenty more. The files that don't appear have an attribute (I) that makes them invisible to the Finder.

You can inspect this metadata with *GetFileInfo* and set it with *SetFile*, both of which are located in */Developer/Tools*:

```
$ GetFileInfo Sample.doc
file: "/Users/bjepson/Desktop/Sample.doc"
type: "W8BN"
creator: "MSWD"
attributes: avbstclinmedz
created: 02/27/2005 11:20:47
modified: 02/27/2005 11:21:01
```

An uppercase attribute indicates that the attribute is toggled on, and lowercase means it is off. The *SetFile* manpage describes all these attributes. For example, to hide the file extension in the Finder, set the E (extension is hidden) attribute:

```
$ SetFile -a E Sample.doc
```

You can also change the creator code and file type with *-c* and *-t*, respectively. For example, to change *Sample.doc* so it's owned by NeoOffice/J (<http://www.neooffice.org>), set the creator code to the empty string and the file type to NO%F, as shown in the following listing and in Figure 2-3.

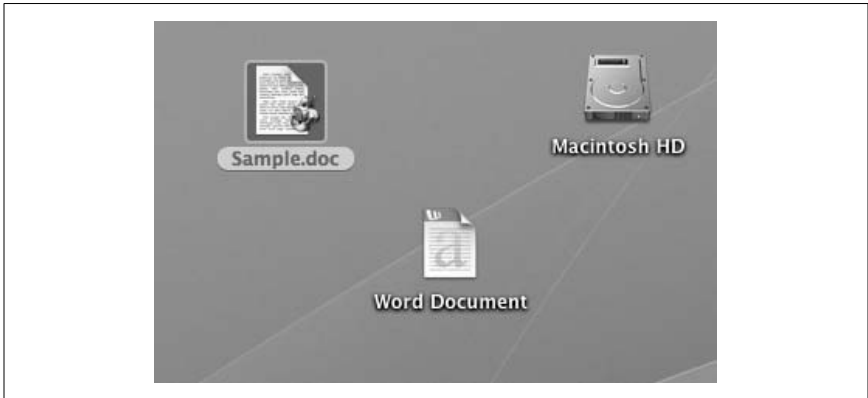


Figure 2-3. NeoOffice owns *Sample.doc* now

```
$ SetFile -c "" -t "NO%F" Sample.doc
$ GetFileInfo Sample.doc
file: "/Users/bjepson/Desktop/Sample.doc"
type: "NO%F"
creator: ""
attributes: avbstclinEdz
created: 02/27/2005 11:20:47
modified: 02/27/2005 11:21:01
```

## UFS

Although UFS doesn't natively support resource forks or HFS+ attributes, Mac OS X finds a place to stash that info. If the file has either a resource fork or any attributes that depend on HFS+ semantics, this information goes into a file named *.\_filename*, where *filename* is the name of the original file (this is known as the Apple Double format):

```
$ touch Foo
$ ls -al
total 4
drwxr-xr-x  2 bjepson  bjepson  1024 Feb 21 20:54 .
```

```

drwxr-xr-x  6 bjepson  bjepson  1024 Feb 21 20:53 ..
-rw-r--r--  1 bjepson  bjepson    0 Feb 21 20:54 Foo
$ SetFile -a S Foo
$ ls -al
total 6
drwxr-xr-x  2 bjepson  bjepson  1024 Feb 21 20:54 .
drwxr-xr-x  6 bjepson  bjepson  1024 Feb 21 20:53 ..
-rw-r--r--  1 bjepson  bjepson   82 Feb 21 20:54 ._Foo
-rw-r--r--  1 bjepson  bjepson    0 Feb 21 20:54 Foo

```

## Preserving Metadata

Before Mac OS X Tiger, you had to be very careful with what you did at the command line. If you used *cp*, *mv*, *rsync*, or any of the other command line utilities that move stuff around, you could have lost part of your file. It was easy to miss this sort of mayhem, since this metadata isn't apparent until you go looking for it, and it wasn't always a disaster. For example, you could have copied a graphics file that kept its preview in its resource fork, and you probably wouldn't have missed it—after all, the next time you opened the image, the application most likely regenerated the preview. But with other files, such as text clippings and Internet locations (drag a URL from Safari to the Finder to create one of these), you lost everything, since all of these files' contents are contained in the resource fork. Here's how it would go on Mac OS X 10.3 and earlier:

```

$ ls -l Resource\Fork\ Example.webloc
-rw-r--r--  1 bjepson  bjepson  0 Feb 21 15:54 Resource Fork Example.webloc
$ ls -l Resource\Fork\ Example.webloc/rsrc
-rw-r--r--  1 bjepson  bjepson  624 Feb 21 15:54 Resource Fork Example.
webloc/rsrc
$ cp Resource\Fork\ Example.webloc foo.webloc
$ ls -l foo.webloc
-rw-r--r--  1 bjepson  bjepson  0 Feb 26 23:18 foo.webloc
$ ls -l foo.webloc/rsrc
-rw-r--r--  1 bjepson  bjepson  0 Feb 26 23:18 foo.webloc/rsrc

```

It's not just the resource fork that got clobbered, you also lost the HFS+ metadata:

```

[Before...]
$ GetFileInfo Resource\Fork\ Example.webloc
file: "/Users/bjepson/Desktop/Resource Fork Example.webloc"
type: "ilht"
creator: "MACS"
attributes: avbstclimEdz
created: 02/26/2005 23:18:33
modified: 02/26/2005 23:18:33

```

```

[After...]
$ GetFileInfo foo.webloc

```

```
file: "/Users/bjepson/Desktop/foo.webloc"
type: ""
creator: ""
attributes: avbstclinmedz
created: 02/26/2005 23:18:52
modified: 02/26/2005 23:18:52
```

Fortunately, Mac OS X Tiger finally makes this problem (mostly) go away by making all the *cp*, *mv*, and *rsync* command-line utilities aware of the resource forks and HFS+:

```
$ cp Resource\ Fork\ Example.webloc foo.webloc
$ ls -l foo.webloc/rsrc
-rw-r--r--  1 bjepson  bjepson  717 Feb 26 23:25 foo.webloc/rsrc
$ GetFileInfo foo.webloc
file: "/Users/bjepson/Desktop/foo.webloc"
type: "ilht"
creator: "MACS"
attributes: avbstclinmEdz
created: 02/26/2005 23:25:21
modified: 02/26/2005 23:27:15
```

If you copy or move the file to a non-Mac system such as a FAT-formatted memory card, Apple Double comes in to save the day:

```
$ cp Resource\ Fork\ Example.webloc /Volumes/NO\ NAME/
$ ls -al /Volumes/NO\ NAME/ | grep Res
-rwxrwxrwx  1 bjepson  bjepson  4434 Feb 27 09:45 ._Resource Fork Example.
webloc
-rwxrwxrwx  1 bjepson  bjepson      0 Feb 27 09:45 Resource Fork Example.
webloc
```

And if you *rm* a file on a volume that's using Apple Double (including UFS as well), it cleans up the *.\_* file:

```
$ rm /Volumes/NO\ NAME/Resource\ Fork\ Example.webloc
remove /Volumes/NO NAME/Resource Fork Example.webloc? y
$ ls -al /Volumes/NO\ NAME/ | grep Res
[... no results ...]
```

For the most part, Mac OS X has you covered when it comes to preserving resource forks. There are a few gotchas that you need to watch out for: *sftp*, *ftp*, and *scp* won't create the resource fork for you.

Also, some tools, such as the Unison File Synchronizer (<http://www.cis.upenn.edu/~bcpierce/unison/>), will try to create the resource forks on the Unix, Linux, or Windows end of the transaction. While this sort of thing works smoothly for the most part, it can occasionally trip you up. We'll talk about those issues and many others in the next chapter.