Spotlight on search: Search and you might be lucky

In the previous article, I explained how Spotlight builds and maintains its indexes of the files on volumes. What I didn't examine was exactly what goes into those indexes, nor how to search them.

The index

Content extracted from each file that's indexed by an `mdworker` process includes:

- file attributes, such as datestamps;
- extended attributes, stored in the file system metadata; these include keywords and copyright information, where provided;
- structured metadata from the main data in the file, as specified by that file type's `mdimporter` plugin; examples include EXIF data;
- content, normally text, exported from the main data of the file, again using the `mdimporter` plugin.

The quality of the data indexed is dependent on that in the file, and its `mdimporter` plugin. Apps which don't provide any plugin to handle their proprietary files make it impossible to use Spotlight to search for their content, and buggy plugins can not only deliver corrupt data for indexing, but can cause `mdworker` processes to crash repeatedly, which may have impact on your Mac's performance.

PDF documents merit special consideration, in particular those which contain scanned pages of printed documents together with OCRed text overlaid on them. Although these may appear perfectly readable, the text extracted from them often contains errors, such as *cftem* for *often,* which makes searching them a nightmare. There is no simple solution to this, as Spotlight's index conforms to the general principle of 'rubbish in, rubbish out'.

Apple hasn't released details of the structure of data within Spotlight's indexes, but it's keyed, with cross-indexing for economy of effort in searching. For example, each file indexed has a type, which can be accessed using the (search) key kMDItemContentTypeTree. Values for the type are UTIs such as public.text. Cross-indexing saves Spotlight from having to search through every file checking its type to discover which contain the desired UTI for any given search. Data is contained in a few large .db files, and a larger number of indexes, maps, shadows, and more.

Search

At its most basic level in macOS, each search is submitted as an MDQuery to the Spotlight server `mds`, which performs its search asynchronously and posts notifications of its progress as it gathers the results. Few apps work at that level, though: for most the preferred interface is the NSMetadataQuery class, which accepts scopes and predicates (of NSPredicate class) which determine the results.

Predicates are structured filters, containing a set of conditions combined with logical operators such as AND and OR, which are common to the search predicates used when obtaining unified log extracts using the `log` command, although each use of predicates has slightly different format conventions. Apple's Predicate Programming Guide gives a comprehensive account, although that document has now been archived, and hasn't been updated for over six years.

Examples of real predicates submitted through NSMetadataQuery are:

```
(** == "syzygy*"cdw) && (kMDItemContentTypeTree == "public.text"cd)
```

which searches for words starting with *syzygy,* case and diacritic insensitive, in files whose UTI is public.text, i.e. plain text files.

```
(kMDItemContentTypeTree == "com.apple.application"cd &&
kMDItemExecutableArchitectures == "*arm*"cd)
```

searches for items whose UTI is com.apple.application, which are applications, and include among their executable architectures the characters *arm,* in other words have executables which run native on Apple Silicon Macs. syzygy999b

In those predicates, terms like *kMDItemContentTypeTree* and *kMDItemExecutableArchitectures* match keys used within the Spotlight database, and are formally known as **Item Attribute Keys**. There are currently over 150 of these for general use, with additional collections for iCloud attributes, and they're listed here. A different list, including custom Item Attribute Keys, is given by

```
mdimport -X
```

which prints the current schema file. That's huge, and broken down by UTI, so is best saved in a file for browsing later. You can also list the metadata attributes for a specific file using

```
mdls filename
```

User interface

macOS provides three public interfaces to Spotlight search:

- Spotlight Search in the menu bar,
- the Finder’s Search window,
- the `mdfind` command tool in Terminal.

Spotlight search in the menu bar is quick and easy, and unsuitable for anything more than casual use, particularly when performing local searches of large numbers of files.

Big Sur has dumbed down the Finder’s Search window too, which used to be readily accessible through the **Find** command. Although it can be used to build refined searches using multiple criteria, there’s no way of saving a default search window, so each use has to be built from scratch. It does give extensive access to Item Attribute Keys, although they are listed using different names which can be confusing. For most users, this is now more frustrating than it is useful.

The `mdfind` command is the complete opposite: although it does support some simple queries, it’s primarily intended to work with predicates as arguments. For example, to send the first of the two example predicates for Spotlight to search, use

```
mdfind "(** == 'syzygy*'cdw) && (kMDItemContentTypeTree == 'public.text'cd)"
```

noting the use of different types of quotation marks for the inner strings and the whole predicate.

Although not without its uses, `mdfind` is too arcane for most of those who need something better than the Finder’s Search window. In the next article, I’ll look in detail at one solution.

Share this: