Going beyond the hierarchical file system: a new approach to document storage and retrieval

Author: Manuel Arriaga
Email: manuelarriaga@fastmail.fm
Copyright 2003 Manuel Arriaga

Introduction

Back in the early days of computing, someone decided that the "natural" way to organize documents stored in electronic format was in folders. Users would create a tree of directories and subdirectories, and documents would go into those folders. The combination of a document's "path" and its file name uniquely identified a document, and constituted the only practical way to store that document's metadata[1].

After more than three decades, the single most important aspect of computing – if you regard computers as machines essentially intended for document creation, manipulation, storage and access – remains unchanged. What was originally a useful analogy ("let us organize electronic documents like we did in paper-based offices, putting documents into folders") became the universal system to store and retrieve documents in electronic format: the hierarchical file system (HFS) has been *the* way to organize documents in a computer throughout computing history[2], and virtually no computer users have ever stored a document using an alternative method.

The supremacy of the HFS is only more surprising if you pause for a second to consider the number of limitations which it imposes on the user. In its attempt to reproduce the familiar paper-based office, the HFS imported a major restriction which is inherent to the real, physical world but which becomes utterly artificial when it is translated into the digital realm: why can't a document (or a subfolder) be stored in more than one folder[3]?

Two other important problems arise from the fact that the user can only store document metadata in one place: the file's path and name. For that reason we end up "encoding" all kinds of unlabeled information into the file's path. Consider, e.g., the path

```
/articles/NYT/2002/Friedman/Stiglitz.html
```

When storing this document the user meant that its document type was "article", that it had been published on the "New York Times", that it had been printed on the year 2002, that its author was called Friedman, that it was about Stiglitz and that it was stored in the file format HTML. However, an important part of that information was lost when it was "encoded" into a path: just as "articles/NYT" might mean both "articles about the New York Times" as well as "articles published on the New York Times", it certainly isn't clear whether this is an article on Stiglitz written by Friedman or the other way around (or possibly an article about both of them written by a third party). Similarly, "2002" can take any number of meanings[4]. Simply put, unlabeled metadata is much less useful than labeled metadata.

---

[1] Other than the one automatically recorded by the operating system, such as creation date, file owner, etc…

[2] The only exception being corporate-level content management systems.

[3] Although all major operating systems offer some kind of soft link ("symbolic links" in Unix and GNU/Linux, "aliases" in MacOS and "shortcuts" in MS Windows), it seems rather unlikely that a significative number of computer users makes extensive use of this feature for the purpose of systematic document organization. I would venture saying that, if these soft links are used at all, they are created with the purpose of making often-accessed documents easily accessible from the "desktop". Not only is the operation of populating a HFS with multiple soft links to a given document awkward, but the result isn't very robust, either: simply renaming a parent directory results in a "dangling" link. So, the question remains: why can't a document/subfolder be (easily) stored in multiple folders?

[4] It can be argued that users might just as well encode the metadata labels into the path. However, just as soft links are rarely if ever used to systematically organize collections of documents, I don't believe many humans create paths such as

```
/type:articles/published:NYT/year:2002/about:Afghanistan-Iraq.kwd
```

on a daily basis.

The fact that all sorts of metadata go into a file's path and name has got another undesirable consequence: the hierarchical "parent -> child" relationships become meaningless. No longer can you look at a path /A/B/C/D and say that D will still have any logical connection with A. (Think of the example above: why is "Friedman" a subdirectory of "articles", or why is "Friedman" a subdirectory of "2002"? From a conceptual point of view this doesn't make any sense: we expect hierarchical relationships to be "transitive".) As a result, whenever a user descends a directory tree she must be permanently trying to guess how to interpret the "parent -> child" relationships at the next level, and that makes HFS browsing unnecessarily difficult.

Finally, storing documents in a HFS according to a certain criterion makes it extremely difficult to later on retrieve them using a different one. Suppose that the user in the example above had initially decided that the first thing to record about each document was its type, the second its source and only then its subject. She would divide them into "/articles", "/photos", etc..., and then at a second level into /(...)/NYT, /(...)/Washington Post, etc... Later on, however, if the user needed to retrieve all documents (irrespective of type and source) that somehow dealt with Joseph Stiglitz she would have to manually browse her entire document collection, going through all subdirectories, to identify the matching documents[5]. The HFS "ties" the future behavior of the user to her original decision on how to lay out the directory tree.

So what is Newdocms?

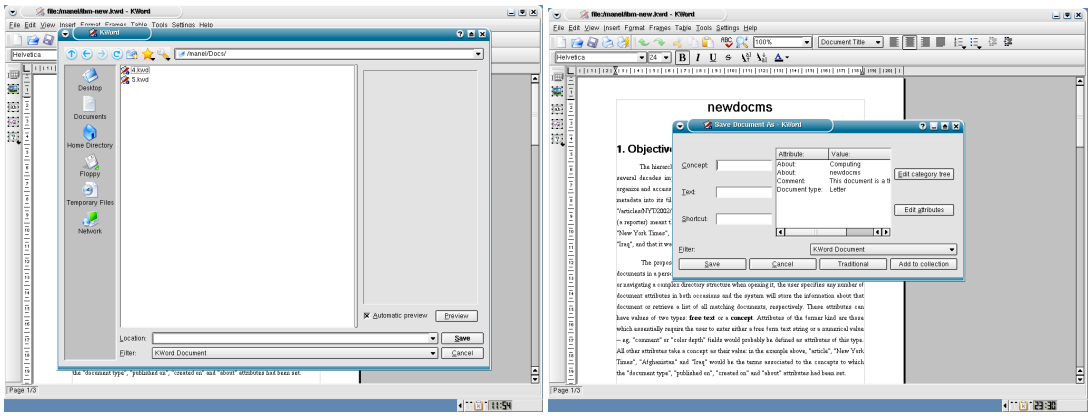

Figure 1: A traditional "Save as" dialog box and the one used in Newdocms

Newdocms aims to do away with these artificial restrictions by insulating the user (as far as possible) from the HFS and allowing her to store document metadata as (attribute,value) pairs. But the importance of hierarchical relationships between concepts has not been forgotten, and its usefulness is now greater than ever.

Newdocms allows the user to set any number of attributes for each document (see Figure 1). These attributes can be of two distinct types: they can be either concept or text attributes. The latter can hold any text string (or number) as a value; typical cases would be "comment", "description" and "color depth". Concept attributes are much more interesting, and they allow the user to define a relationship between the document in question and a given concept.

The easiest example probably is that of the "about" attribute: a document can be "about cats", or "about politics", e.g. (In these cases the concepts would be "cats" and "politics"). In a similar way, a (document that is a) short story can be described as "taking place in India", or "written by Rudyard Kipling". (Here the relevant concepts would be "India" and "Rudyard Kipling"). Of course, the user only needs to set a certain attribute for a given document if it makes sense: she probably wouldn't care about the "color depth" of a text document.

The most interesting part is that the user can organize all concepts into a category tree. This is just a typical hierarchical structure, but since there isn't any need to put any document metadata into the

[5] Again, some will argue that it is possible to use software tools to search a HFS for files with certain expressions in their paths/names. However, like I said before, the fact that it is possible to do so means neither that (i) doing so is practical nor that (ii) it is useful to the average computer user.

Copyright (©) 2002 Manuel Arriaga

hierarchy (unlike what happened when using the HFS) the "parent -> child" relationships can be read as "(child) is a (parent)". This allows Newdocms to gain a basic form of ontological intelligence: it "knows" that, if both "dogs" and "cats" are subcategories of "animals", then any reference to "animals" should be interpreted as referring not only to "animals" in general but should also encompass specific references to "dogs" and "cats". The result is that you can be as specific as you wish to when describing a document ("this document is about black, friendly dogs") because you know that later on searching for a higher level concept ("show me all documents about animals") will bring you all relevant matches, even if those documents weren't specifically defined as relating to that (higher level) concept.

Another powerful feature of this system lies in the fact that the same concept can be represented by multiple categories. Since our conceptual representation of reality isn't unidimensional (we are able to sort things by different criteria; e.g., a person is not only of "British nationality" but also "a good friend" and "relatively short"), the possibility of arriving at the same concept by taking different logical paths can be very useful: e.g., asking for documents "written by North-Americans" and asking for documents "written by professors of literature" would both lead to a list of matching documents which included texts by Harold Bloom (assuming that you had catalogued Harold Bloom both as a North-American (nationality) as well as a professor of literature (occupation))[6].

My goal with Newdocms is to make large (personal) collections of electronic documents manageable and to allow for powerful and easy document retrieval. In addition to serving this "long-term archival" objective, Newdocms also includes a handy feature intended to make the daily retrieval of often-accessed documents easier. That feature consists in the definition of document "shortcuts": simple text strings which, when entered into the appropriate text field, bring up the associated document(s) in an easy and immediate way. The query results are always sorted by last access time, meaning that the most recently viewed matching document will come at the top of the list, and will, therefore, be easier to select.

A third alternative when retrieving a document is to simply press "Open" on the dialog without entering any attribute (nor shortcut): in that case you will simply get a list of all documents sorted chronologically.

When saving you don't need to specify any attributes: if you are in a hurry simply press "Save" and the document will be saved and marked as "uncatalogued" (you can always access it through the chronological list of documents described in the previous paragraph). Later on, when you have more time, you can come back to it and catalogue it.

It is important to notice that Newdocms doesn't require “more work” (when saving documents) than the traditional HFS: you tell Newdocms as much (or as little) as you wish to about the documents you save – as you have just seen, you can even provide zero information about a document and still save it! (Of course, the more information you provide when saving documents the more accurate the query results will be.) The only absolutely necessary information when saving is a file format; if the application doesn't specify one you will be requested to enter it manually in a text line edit, either as a MIME type or as a file name pattern (e.g., "*.txt").

Another feature of Newdocms is its support for what I call "collections": these are sets of files which share the same attributes, and are regarded by Newdocms as a single document. This might be useful, e.g., for digital photos: you can have 50 pictures taken at the beach in July featuring your girl-friend, her dog and some friends. You probably wouldn't feel like manually defining those same attributes 50 times, so you can simply create a new collection (called "Photos at the beach 2002/07") for which you define those attributes only once, and then insert the pictures into the collection. When inserting a file into a collection you can either provide a file name or simply point Newdocms to a directory holding your pictures and tell it to “import” all those files into the collection.

When a collection matches the attributes you defined, it is identified as a collection in the "Matching documents" window. Only collections which (i) match the attributes you specified and (ii) hold files which the application is able to read are listed there. If the file you are looking for is inside one of the matching collections, simply select that collection and you will be given the opportunity to (through a traditional "Open" dialog pointed at the contents of that collection) select the desired file.

If things go wrong, Newdocms will always try to present to the user a traditional (HFS) "Open" or "Save as" dialog so that the file can be safely saved somewhere. Or, if you need to open or save a document outside of Newdocms simply hit the "Traditional" button and the familiar file dialog will appear. By installing Newdocms you are not forcing yourself to use it every time you open/save a file.

[6] This is the purpose of conceptless (or "null concept") categories: they act simply as container categories which specify the criterion used by its subcategories. Examples of conceptless categories would be "People -> By nationality" and "People -> By occupation". Of course, the children categories of conceptless categories can (and should!) be associated to concepts.

So where do your documents go when you save them with Newdocms? As you might have noticed (if you looked at the window titles after saving something), they are stored as ~/Docs/{numeric id}.{ext}[7]. All the metadata is stored in a file called ~/newdocms.db. In that file each document's attributes are associated with its unique numeric id (the one which is used as a file name).

Notice that the category tree, although being a hierarchy just like the traditional HFS, doesn't exist as a directory tree on disk: it only exists as a data structure inside the database file. All documents are saved in a "flat store" (under ~/Docs/), and they are associated to the relevant concepts inside the database file.

In order to provide a meaningful path to your documents (chiefly so that looking at window titles allows you to know which document that window is displaying – this feature is not meant to have you browsing your home directory in the traditional way), when you open a document an artificial path (formed by a concatenation of the concepts to which that document is associated) is created in ~/View and “symlinked” to the real file (which resides in ~/Docs). It is advisable that before running Newdocms for the first time you make sure that no directories with those names exist in your home directory.

A step-by-step example of how it works

First you save a document, defining its attributes and specifying its file format. In Figure 2 I am saving a Salon article on Joe Strummer, the leader of the British punk band The Clash.

Figure 2: Saving a document in Newdocms

When specifying the attributes of a new document you might need to define new concepts/categories. Simply press the "Edit category tree" button in the "Save as" dialog and then – as seen in Figure 3 – create new categories by right-clicking and choosing "Add category" from the pop-up menu.

[7] Collections are simply stored as directories called ~/Docs/{numeric id}.

Copyright (©) 2002 Manuel Arriaga

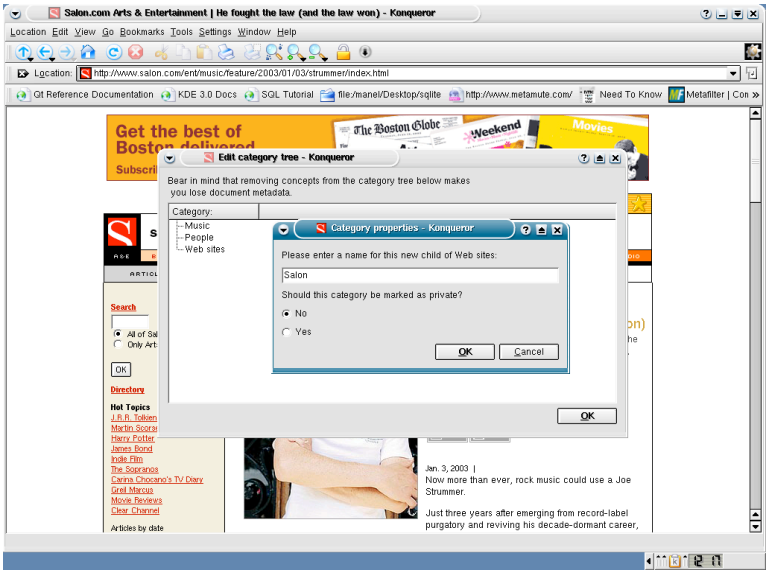

Figure 3: Adding categories to the category tree

Suppose that later you wish to open a document about that band, irrespective of the file format it is stored in. By specifying those two attributes in the dialog box displayed in Figure 4, you perform a query on the Newdocms database.

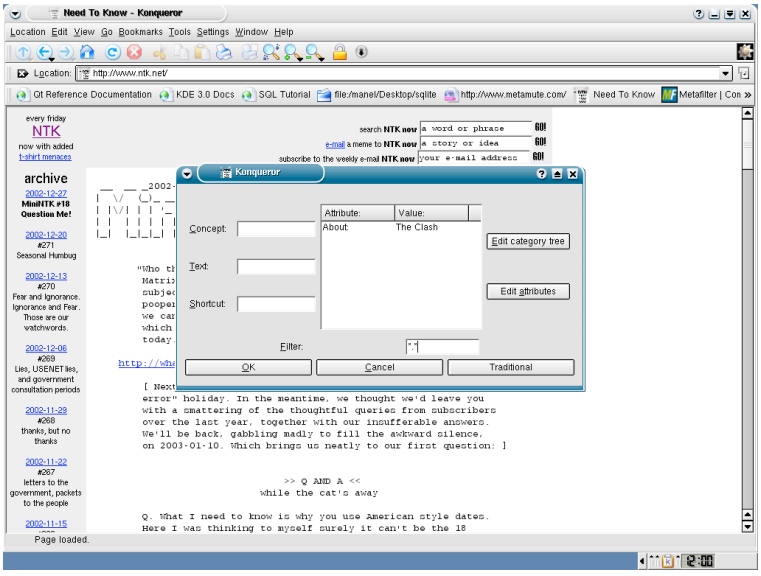

Figure 4: Dialog box used when making queries

The result of that query is a list of matching documents. In that dialog you select the document which interests you, click "Ok" and that document will be opened by the application. That is done in Figure 5.

Copyright (©) 2002 Manuel Arriaga

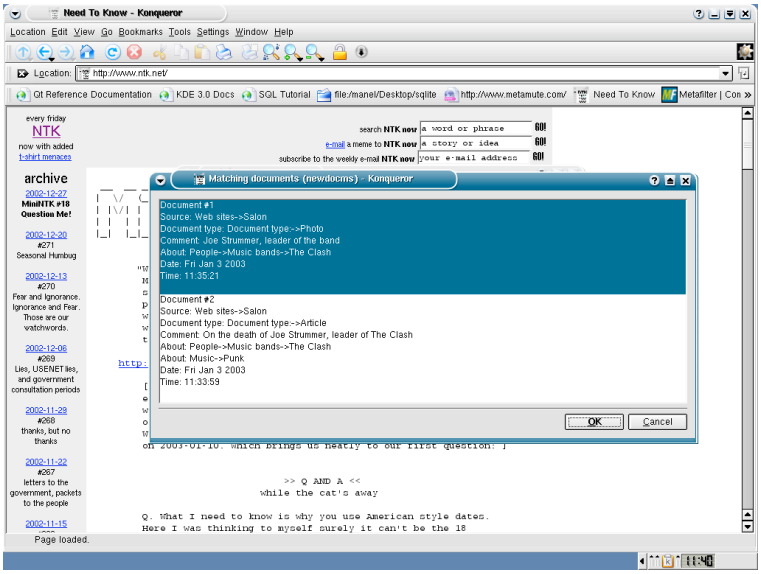

Figure 5: A dialog box listing the matching documents

Conclusion

This is a very early preview version of Newdocms, a project I have been working on for two years[8]. I honestly think that this is a truly innovative system, and that something both powerful and useful can be made out of this. At the time being it only works with KDE, but I would also like to modify the GNOME and OpenOffice.org libraries (any volunteers?[9]). As time permits, I will create a stand-alone application (a sort of document "browser") that will allow access to the documents from outside specific applications.

The user interface, in its current form, is neither friendly nor "sexy": using a line edit (with auto-completion) to specify the concept is just an idea I had (and personally like), but it would be very easy to radically change the user interface. The system is completely modular and the way in which the user defines document attributes depends on a single class; nothing else would need to be changed.

The possibilities are endless: one could select concepts directly from the category tree instead, or be presented with an iconic representation of concepts and their relationships... All this would require minimum effort, because the rest of the system would remain unaltered.

I am not sure most people (even among those who deal with free software) have grasped the huge number of possibilities which this software ecosystem presents us with: starting with the work already done by thousands of coders world-wide, we can, with a little bit of imagination and some work, create amazing things.

In a sense, Newdocms is a tribute to the free software nature of the underlying systems (KDE, GNOME, OpenOffice.org and, more generally, GNU/Linux): I had an ambitious idea, but I would have never been able to write from scratch a complete desktop environment with dozens of fully-developed applications to implement it in a usable way. Because these systems are free software, it was possible for me – from both a legal and technical point of view – to modify them and to make something truly original that otherwise would never exist. The next time you hear someone comment how "free software is killing innovation" you might wish to tell him this story.

[8] You might notice, besides many other things I hope you will tell me about, that selecting "Save link as" in Konqueror when the pointer is over a link to a Perl CGI-generated web page will make Newdocms tell you that that document will be catalogued as "Perl source code"… Just use the traditional file dialog for now, save it with a name ending in ".html", reopen it in Konqueror and choose "Save as". Other minor problems include the unusual tab order in the dialogs and the keyboard shortcuts in the main dialog.

[9] Very little work would be involved: only the file selector of the underlying system would need to be reimplemented. The core of Newdocms would remain unchanged. If you feel like giving it a try, don't hesitate to contact me for any additional information.

Copyright (©) 2002 Manuel Arriaga