9

An Introductory Tour of SMB

> The devil is in the details.
>
> — Popular saying

We will start with a quick museum tour of SMB. Our guide will be the venerable **U**niversal **N**aming **C**onvention (UNC). You may remember UNC from the brief introduction way back in Chapter 1 on page 3. UNC will provide directions and point out highlights along the tour.

Please stay together, everyone.

The UNC directions are presented in terms of a path, much like the **U**niform **R**esource **I**dentifier (URI) paths that are used on the World Wide Web. To explain UNC, let us first consider something more modern and familiar:

```
http://ubiqx.org/cifs/index.html
```

That string is in *URI* syntax, as used by web browsers, and it breaks down to provide these landmarks:

```
      http == The protocol to use.
ubiqx.org == The name of the server.
      cifs == The directory path.
  SMB.html == The file name.
```

The landmarks guide us along a path which eventually leads us to the file we wanted to access.

The SMB protocol pre-dates the use of URIs and was originally designed for use on LANs, not internetworks, so it naturally has a different (though surprisingly similar) way of specifying paths. A **U**niversal **N**aming **C**onvention (UNC) path comparable to the URI path above might look something like this:

```
\\ubiqx\cifs\SMB.html
```

...and would parse out like this:

```
  ubiqx == The name of the server.
   cifs == The directory path.
SMB.html == The file name.
```

Very similar indeed.

One obvious difference between the two formats is that UNC doesn't provide a protocol specification. That's *not* because it always assumes SMB. The UNC format can support all sorts of filesharing protocols, but it is up to the underlying operating system or application to try to figure out which one to use. Protocol and transport discovery are handled by trial-and-error, with each possibility tested until something works. As you might imagine, a system with AppleTalk, NetWare, and SMB all enabled may have a lot of work to do.

The UNC format is handled natively by Microsoft & IBM's extended family of operating systems: DOS, OS/2, and Windows.[1] Samba's `smbclient` utility can also parse UNC names, but it does so at the application level rather than within the OS, and it only ever tries to deal with SMB. Even so, `smbclient` must handle both NBT and naked transport, which can be tricky.

9.1 The Server Identifier

The first stop on our UNC tour of SMB is the server name field, which is really a server *identifier* field because it will accept addresses in addition to names. This book concerns itself with only two transports — NBT and naked TCP transport — so the only identifiers we care about are:

---

1. Steve French says that OS/2 may have been the first OS to fully support the UNC scheme.

- NetBIOS names,
- DNS names, and
- IP addresses.

NetBIOS and DNS names both resolve to IP addresses, so all three are equivalent.

Sort of...

Recall that the NBT `SESSION REQUEST` packet requires a `CALLED NAME` in order to set up an NBT session with the server. Without a correct `CALLED NAME`, the NBT `SESSION REQUEST` *may* be rejected (different implementations behave differently). So...

- if the transport is NBT (not raw),
- and the server is identified using a DNS name or IP address...

...then we're in a bit of a pickle. How do we find the correct NetBIOS name to put into the `CALLED NAME` field? There really is no "right" way to reverse-map an IP address to a particular NetBIOS service name. The solution to this problem involves some guessing, and it's not pretty. We will go into detail when we discuss the interface between SMB and the transport layer.

Of course, if SMB is running over raw transport then there is no NBT `SESSION REQUEST` message and, therefore, no `CALLED NAME`. In that case, the NetBIOS name isn't needed at all, which saves a lot of fuss and bother.

9.2 The Directory Path

The directory path looks just like a directory path, but there is one small thing that makes it different. That thing is called the "share name."

Whenever a resource is made available (shared) via SMB it is given a share name. The share name doesn't need to be the same as the actual name of the object being shared as it exists on the server. For example, consider the directory path below:

```
/dogs/corgi/stories/jolyon/
```

Suppose we just want to share the `/stories` subdirectory. If we simply call it "`stories`" no one will know what kind of stories it contains, so we

should give it a more descriptive name. We might, for example, call it "`dogbytes`".

The share name takes the place of the actual directory name when the share is accessed via SMB. If the server is named "`petserver`", then the UNC path to the same directory would be:

```
\\petserver\dogbytes\jolyon\
```

As shown in Figure 9.1, there can be more than one share name pointing to the same directory and access rules may be applied on a per-share basis. The idea is similar, in some ways, to that of symbolic links (symlinks) in Unix, or shortcuts in Windows. The share is a named pointer — with its own set of attributes — to the object being made available by the server.

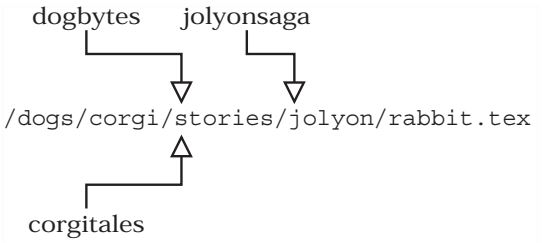

Figure 9.1: *SMB shares*

Share names are similar to Unix `symlinks`. Multiple share names may point to the same directory on the server or to different directories. Each share may have its own set of permissions.

9.3 The File

This is the last stop on our quick UNC tour of SMB.

Files, like directories, should be fairly familiar and fairly straightforward. As has been continually demonstrated, however, things in the CIFS world are not always as simple as they ought to be. Our point of interest on this part of the tour is the distinction between server filesystem syntax and semantics — and client expectations... a very gnarled knot for CIFS implementors.

Consider, for example, a bunch of Windows clients connecting to an SMB server running on Linux. On the Linux system the filenames `Corgi`,

`corgi`, and `CORGI` would all be distinct because Linux filesystems are typically case-sensitive. Windows, however, expects filenames to be case-insensitive, so all three names are the same from the Windows point of view. Thus, we have a conflict. How does a Linux server make all three files available to the Windows client?

Other difficult issues include:

- filename lengths,
- valid characters,
- file access permissions, and
- the end-of-line delimiter in text files.

These are complex problems, not easily solved. The CIFS protocol suite is *not* designed to be agnostic with regard to such things. In fact, CIFS goes out of its way at times to support features that are specific to DOS, OS/2, and Windows.

...and that concludes our tour. It's time to visit the gift shoppe.

9.4 The SMB URL

The UNC format is specific to one family of operating systems. Earlier on, though, we compared UNC with the more portable and modern URI format. That's called foreshadowing. It's a literary trick used to build suspense and anticipation.

There is, in fact, such a thing as an SMB URL. It fits into the general URI syntax[2] and can be used to specify files, directories, and other SMB-shared stuff. It is intended as a more portable and more complete way to specify SMB paths at the application level.

As of this writing, the SMB URL is only documented in an IETF Internet Draft, and is not yet any kind of standard. That hasn't stopped folks from implementing it, though. The SMB URL is supported in a wide variety of products including the KDE and GNOME desktop GUI environments, web browsers such as Galeon and Konqueror, and Open Source CIFS projects like

2. The distinction between a URL and a URI is subtle, and confuses me to no end. Fortunately, it is not something we need to worry about.

jCIFS and `libsmbclient` (the latter is included with Samba). Thursby Software and Apple Computer also make use of the SMB URL in their commercial CIFS implementations.

That's good news for CIFS implementors because it means that there is an accepted, cross-platform way to identify SMB-shared resources, both within LANs and across the Internet.

9.5 Was That Trip Really Necessary?

Our quick UNC tour provided an introduction to some of the basic concepts — and annoyances — of SMB. We will expand upon those ideas as we dig more deeply into the protocol. The UNC format itself is also important for a variety of reasons, both historical and practical. Not least among these is that UNC strings are used within some of the SMB messages that cross the wire.

The SMB URL format is equally significant. It is portable, flexible, and gaining in popularity. It will also form the basis for examples given later in the text. If you are implementing an SMB client, you will most likely want to have some convention for identifying resources. You could invent your own, or use UNC, but the SMB URL is probably your best option.