Towards full NTFS semantics in Samba

Andrew Tridgell
tridge@samba.org

About Samba

- Started in 1991 as a side project in my spare time
- Now have about 25 "Samba Team" members
- Ported to a wide variety of OSes
- Massive user base now built up (millions of installations?). Over 25 books published.
- Sometimes referred to as the 'stealth weapon' of the Linux community
- Developed using analysis of network traces
- Currently about 350k lines of code
- Used in many commercial products, especially NAS

Semantic Conversion

- One of the major challenges facing NAS boxes is the problem of 'Semantic Conversion'
- Each of the protocols that the box supports has a quite different set of semantics, and these semantics are usually quite different from the native semantics of the local operating system (often Linux)
- For 'correct' operation the server must map the expected protocol semantics onto the semantics of the local OS, and this mapping must be fast!

Semantic Conversion : pt 2

- The main semantic conversion problems are
  - locking
  - ACLs
  - case insensitivity
  - short/long names
  - delete/rename
- There are two broad approaches to each of these. Either map the required semantics onto the local OS semantics or add 'parallel semantics' into the local OS

Byte range locking

- A good example of semantic conversion problems is byte range locking.
  - POSIX locks are not stackable, CIFS are
  - POSIX locks have a fixed lock context
  - signed/unsigned lock offsets
  - POSIX locks are advisory, CIFS mandatory
- Is byte 6 locked after this sequence?
  - lock(1,10)
  - lock(5,8)
  - unlock(1,10)

Case insensitivity

- Applications running on CIFS clients expect file systems to be case insensitive, whereas Unix systems are case sensitive. How do you provide case insensitive semantics on a case sensitive operating system?
- Lets walk through the worst case - how do you prove that the file /home/test/data/test.dat doesn't exist?

Case insensitivity - current method

- To prove that /home/test/data/test.dat doesn't exist you need to:
  - open '/' and search for names that match "home"
  - open '/home' and search for names that match "test"
  - open '/home/test' and search for "data"
  - open '/home/test/data' and search for "test.dat"
- this can cost hundreds of system calls
- Some optimizations for common cases are possible

Case Insensitivity - pt 2

- The alternative is to add case insensitive support directly into the kernel. To do this on Linux you need to modify two main kernel subsystems, the low-level file system and the dcache.
- In the simplest case we need to:
  - change the file system to use strcasecmp() when looking up names in directories
  - change the dcache hash function to be case insensitive
  - change the dcache comparison functions to use strcasecmp()
- Things soon get a bit more complex!

Case Insensitivity - XFS

- The on-disk directory format in XFS is a hash. This means that we need to change the on-disk format when we change to a case insensitive hash.
- For backwards compatibility we need to mark each directory in XFS as being either case insensitive or case sensitive. The directory hash function is then chosen based on this flag.
- Other major problems include:
  - case insensitive in what character set?
  - per-process case insensitivity (for NFS + CIFS)
  - negative dentry problems!

More semantic conversion

- The other major points of semantic mismatch are
  - File ACLs (access control lists)
  - short/long names
  - delete/rename semantics
- With each of these we have the choice of semantic mapping or parallel access. Usually parallel access is preferable, but it is often much more complex to implement

Proposal - a new Samba VFS

- The current Samba VFS allows loadable modules to replace all IO functions at the Posix level
  - used for virus checkers, trash can etc
- The current VFS also contains loadable methods for NT ACLs, but doesn't contain any operations for oplocks, share modes, 8.3 names or case-sensitive handling
- A new Samba VFS system is needed that allows all the CIFS->Posix mappings to be replaced

Move the VFS access points

- The first step is to move the VFS access points much closer to the top of the CIFS stack.
- This means that the VFS entry points will no longer be Posix functions like open() but CIFS functions like NTCreateX().
- This will also greatly reduce the distance between the parsing of a CIFS packet from the network and the VFS entry point.

A POSIX backend

- The next step is to rearrange the existing code to form a new VFS backend based on the current CIFS -> Posix mapping.
- This is needed to keep Samba working while the new VFS is being developed.

A CIFS Backend

- The first backend that I plan on developing is a CIFS->CIFS backend. This will use a remote CIFS server to provide the storage for the VFS backend.
- The idea is to provide an 'ideal' backend to ensure that the new VFS can handle the full range of CIFS semantics.

A reference backend

- Another critical portion of the new VFS will be the creation of the 'reference backend'. This backend will aim to provide close to 100% CIFS/NTFS semantics, but will not attempt to integrate with the OS or be efficient.
- The plan is to store all files with fixed permissions and ownership. Each file will have a corresponding record in a database, with the record containing all the CIFS/NTFS meta data needed for full CIFS semantics. The meta-data will include both static data (like ownership and ACLs) and dynamic data (like oplock state).

Test suites

- One we have the reference backend in place it should be possible to create dual-server test suites that test much finer detailed CIFS compliance with Windows than is currently possible.
- This will give us a basis for validating our CIFS protocol behavior, and will give a good basis for other groups to create a backend that takes advantages of specific attributes of more specialised filesystems.
- The 'dual-server' methodology used in other Samba testsuites will be used

Dual-server testing

- As CIFS lacks a comprehensive protocol specification we use dual-server testing to validate the code
- A dual-server test attaches in parallel to both a reference server (such as Win2000) and a test server (such as Samba)
- The test code does either exhaustive or randomized case generation and looks for any differences in the replies from the two servers
- A binary search back-tracking system is used to find the divergence point

Other major development tasks

- The new VFS is only one small part of what is going on in the Samba world
  - Active Directory integration
  - Domain controller development
  - Internationalization
  - SPOOLSS/printing developments
  - better management tools