SMB3.11 Unix Extensions current status

SambaXP 2025
Göttingen

Volker Lendecke

SerNet / Samba Team

2025-04-07

Why?

- NFS is the "native" Linux file sharing protocol
  - Initial setup simple: Edit /etc/exports on the server and /etc/fstab on the client, and it works.
  - Reasonable compatibility with what Linux applications expect
  - Metadata caching problematic
  - Locking does not work, deleting open files might leave tombstones around
  - Without Kerberos: **N**o **F**ile **S**ecurity
  - Kerberized NFS hard to set up and not bug-free
- SMB comes from the Windows world
  - SMB3.11 is secure by default
  - Cache coherency is solved with Oplocks and Leases
  - Locking works
  - SMB servers already exist almost everywhere, setup equally simple
- **Because we can**

SMB3 Posix Extensions

- Make SMB a competitor to NFS
- Extend SMB with behavior Posix clients expect
- Client can ask for Posix Extensions when connecting
  - New negotiate context
- File Name handling
  - Case Sensitive, no reserved names and streams
  - New Posix Create Context – “xattr” on an API call
- Posix Metadata
  - New file information class
  - permissions, ownership, all of `struct stat`
- https://gitlab.com/samba-team/smb3-posix-spec.git

File types in SUSV4

- Opengroup defines 7 types of files
  - `S_IFREG` Regular file
  - `S_IFDIR` Directory
  - `S_IFBLK` Block device (/dev/sda)
  - `S_IFCHR` Character (/dev/null)
  - `S_IFIFO` FIFO (named pipe)
  - `S_IFLNK` Symbolic link (/etc/alternatives/editor)
  - `S_IFSOCK` Socket (for example d-bus server)
- Regular files and directories are well understood, semantics similar between Posix and SMB
- Differences being taken care of by Samba since it was started
- What about the others?

Samba's role for Posix special files

- Block and Character devices present hardware to user space
  - Opening /dev/null should not send garbage over SMB
  - Device files only make sense locally
- FIFOs and Sockets are a host-based inter process communication mechanism
  - Windows named pipes are an IPC mechanism across the network
  - FIFOs across smbd used to work. they got accidentially broken, but nobody noticed.
  - Homedirs over SMB must present FIFOs and Sockets for local IPC
- Sockets on share IPC$:
  - DCERPC (Domain Controller, Printing, Registry etc)
  - Patches available for Windows Search Protocol, need review
- Symlinks: Hot debates, security problems all over the place
  - Look for Jeremy Allison's talk, delegate handling to clients

NTFS reparse points

- Wikipedia: Reparse points provide a way to extend the NTFS filesystem. A reparse point contains a reparse tag and data that are interpreted by a filesystem filter driver identified by the tag.
- Applications can set an arbitrary blob as a reparse point
- When opening a file, NTFS filters can interpret the contents
- A reparse point not handled by any filter gives STATUS_IO_REPARSE_TAG_NOT_HANDLED
- [MS-FSCC] defines a few dozen reparse tags, most of them as “not meaningful over the wire”
- SMB clients can still access them, “not meaningful over the wire” just means “we won’t document them”

Windows NFS Server

- Once you install the Windows NFS server, the properties of a directory offer “NFS Sharing” next to “Sharing”
- Windows NFS exports normal NTFS files and directories
  - It has to store the NFS special files somewhere
- [MS-FSCC] defines `IO_REPARSE_TAG_NFS` to be used by the NFS server. Also “not meaningful over the wire”, but...
  - 2.1.2.6 defines `NFS_SPECFILE_LNK` and others for `_BLK`, `_CHR`, `_FIFO` and `_SOCK`.
- `_BLK` and `_CHR` have 32-bit major and minor numbers as data
- `_SYMLINK` has the target as Unicode (UTF-16)
- Windows properties show “L” for all reparse points created over NFS

Listing directories

- `ls -l` not only lists files, but also permissions and type
- NFS started with just a readdir RPC call that only lists names, just like the readdir-syscall
- `ls -l` does a stat(1) on each name, which is a roundtrip to the server
- NFS readdirplus returns names plus attributes, avoids roundtrips
- SMB only has readdirplus with infolevels
- Posix extensions add `struct stat` infolevel
- Last year's talk had a slide "WSL vs NFS reparse points"
  - Undocumented WSL avoids roundtrips, file type encoded in readdirplus
  - Documented NFS adds roundtrips to ask for file types
- Solution since last year: Use documented NFS reparse points, define file type as part of `struct stat` permission

Symlinks

- With symlinks, we have 3 options
  - WSL `IO_REPARSE_TAG_LX_SYMLINK`
  - NFS `NFS_SPECFILE_LNK`
  - Native NTFS `IO_REPARSE_TAG_SYMLINK`
- `IO_REPARSE_TAG_SYMLINK` is the only one properly interpreted by the SMB server
- Trying to cross a symlink when opening a file gives `NT_STATUS_STOPPED_ON_SYMLINK`
  - Additional error information shows symlink target
- Samba presents existing symlinks as `IO_REPARSE_TAG_SYMLINK` and returns `NT_STATUS_STOPPED_ON_SYMLINK`

Creating special files over SMB

- Two steps:
  - Just create a file with `OPEN_REPARSE_POINT`
  - Issue `FSCTL_SET_REPARSE_POINT` to set the content blob
- smbd does the same: Create files with `REPARSE_POINT` attribute
  - Security: You don't want to create a block device with 777 permissions
  - Semantics: You can't turn a file atomically into anything else

Long-running compute jobs

- SMB is always authenticated
  - Without username/password (or Kerberos ticket) there's no access
- **N**o **F**ile **S**ecurity helps for compute farms
- Standard SMB3 offers `SMB2_REMOTED_IDENTITY_TREE_CONNECT`
  - A compute node gets a machine account
  - The SMB server marks this node as trusted
  - When the compute node connects to a share *as a machine*, it can transmit a user identity
  - This user identity is trusted by the server, so a compute job can assume a user identity.
- Alternative: Kerberos delegation with gssproxy, search for "Daniel Kobras gssproxy" and watch his talks

Status / Next steps?

- Server code is done for special files
- `NT_STATUS_STOPPED_ON_SYMLINK` returned for posix and `follow symlinks = no`
- Linux 6.13 has code for Posix special files and for symlink handling
- Missing: ACL representation
  - Goal: `ls -l` shall show “+” for files with ACLs
  - getfacl and setfacl shall work
  - A lot of discussion ahead

Thanks for your attention

vl@samba.org / vl@sernet.de
https://www.sernet.de/
https://www.samba.org/