Reparse Points in SMB

Storage Developer Conference 2023
Fremont

Volker Lendecke

Samba Team / SerNet

2023-09-20

SMB3 Posix Extensions

- Make SMB a competitor to NFS
- Extend SMB with behavior Posix clients expect
- Client can ask for Posix Extensions in NegProt request
  - New negotiate context
- File Name handling
  - Case Sensitive, no reserved names and streams
  - New Posix Create Context
- Posix Metadata
  - New file information class
  - permissions, ownership, all of `struct stat`
- This talk is *NOT* about any of this
- https://codeberg.org/SMB3UNIX by David Mulder

File types in SUSV4

- Opengroup defines 7 types of files
  - `S_IFBLK` Block special.
  - `S_IFCHR` Character special.
  - `S_IFIFO` FIFO special.
  - `S_IFREG` Regular.
  - `S_IFDIR` Directory.
  - `S_IFLNK` Symbolic link.
  - `S_IFSOCK` Socket.
- SMB3 in Samba handles `S_IFREG` and `S_DIR` well.
- What about the others?

Samba's role for Posix special files

- Samba has to present special files it finds to clients
  - Normal files and directories are taken care of
- FIFOs are broken right now
  - Clients can open a FIFO, but read/write fails
  - Samba removed `SMB_VFS_READ` and `SMB_VFS_WRITE`
  - FIFOs don't like pread/pwrite
- Sockets only work for named pipe (MS-RPC) servers, such as samba-dcerpcd
- Block and character devices don't make sense over SMB, but should be visible for clients to use locally

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 Subsystem for Linux

- WSL v1 used NTFS to represent Linux special files
- `IO_REPARSE_TAG_AF_UNIX` used for sockets
- `IO_REPARSE_TAG_LX_BLK`, `_CHR` `_FIFO` for the obvious Linux counterparts
- None of them are documented
- WSL v2 uses ext4 on a block device, it does not need NTFS reparse points anymore

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

WSL vs NFS reparse points

- WSL defines distinct reparse tags per type
  - Format is undocumented, although probably not rocket science to find out
- NFS only uses one reparse tag
  - Distinguishes object types within the reparse point contents
- Pro NFS:
  - Documentation available
  - Protocol-Level tests with NFS possible
  - mkfifo over SMB will create a valid entry for NFS to serve a FIFO
- Pro WSL:
  - NFS reparse points require another round-trip when listing a directory
  - `QUERY_DIRECTORY` gives the reparse tag, with WSL that's sufficient for `FIFO`s and `SOCK`s
- My vote: Use NFS reparse tags due to their interop story

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
  - Easy to follow symlinks client-side
- Samba should present existing symlinks as IO_REPARSE_TAG_SYMLINK and return 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

Status / Next steps?

- Most of the server code is in MR2887
- How and when to activate server-side code?
  - Bind `NT_STATUS_STOPPED_ON_SYMLINK` to `follow symlinks = no`?
  - Set `follow symlinks = no` on SMB3 Posix opens?
- How to deal with (currently broken) FIFOs?
  - Always report as reparse points?
  - Other special files?
- Incomplete: Reparse points over SMB1
- Chicken-and-Egg problem: We don't have clients yet
  - Right now working on libsmbclient to include in user-space clients (KIO, gvfs)

Thanks for your attention

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