STORAGE DEVELOPER CONFERENCE
SDC21
BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

A SNIA Event

The Road to the New Samba VFS

Ralph Böhme, Samba Team Lead, SerNet
2021-09-28

Road to a modern VFS for SMB2+

Samba 4.15

Samba 4.15.0 Release Notes

```
NEW FEATURES/CHANGES
====================

VFS
---

The effort to modernize Samba's VFS interface is complete and
Samba 4.15.0 ships with a modernized VFS designed for the post
SMB1 world.
```

Woohoo! :)

The Road to a modern VFS for SMB2+

Samba 4.15 finishes the VFS modernisation

- Ongoing effort since a few years, initially driven by Jeremy Allison
- Standardizing path based filesystem syscalls on `at()` variants
    - eg `openat()` instead of `open()`
- use file handles instead of paths as often as possible
    - eg `fstat()` instead of `stat()`

Why did we do this?

How did we get there?

Some History

SMB1 Fallacies: Pervasive use of Paths

A path by any other name would smell as unpleasing.

Most metadata operation (get and set) in SMB1 can be done on paths:

- Path processing is complex and slow
  - the core function `unix_convert()` had more then 800 lines (before we refactored it last year)
  - plus several thousand lines of code in related helper function

So what's wrong with paths? Things to consider:

- Charset conversion
- Mangling Windows incompatible paths
- DFS paths
- SMB1 previous version paths (with `@GMT-...` tokens in the path)
- Case insensitive semantics
- Named streams support

STORAGE DEVELOPER CONFERENCE SDC 21 By Developers FOR Developers

Enter SMB2

By contrast, SMB2+ is a purely handle based protocol

- `SMB2 Create` request takes a pathname
- Everything else operates on a handle returned by `SMB2 Create`
- . . . with a few exceptions:
  - `QueryInfo(NormalizedNameInformation)` returns a full pathname
  - `QueryDirectory()` returns relative pathnames
  - `SetInfo(File{Link,Rename}Information)` takes a full target pathname

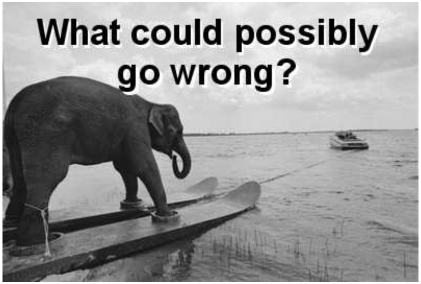
Deprecation of SMB1 in 4.11

- The world has moved away from SMB1
- So did we, SMB1 is now disabled by default
- Not yet removed completely: used in tests

The Idea

The idea: a (mostly) handle-based VFS for the SMB2+ World

- Streamline the VFS interface to be (mostly) handle-based
  - `SMB_VFS_FSTAT()` instead of `SMB_VFS_STAT()`
  - `SMB_VFS_FGETXATTR()` instead of `SMB_VFS_GETXATTR()`
  - `SMB_VFS_FGET_DOS_ATTRIBUTES()` instead of `SMB_VFS_GET_DOS_ATTRIBUTES()`
  - . . . and so on.
- Perfect match for the SMB2+ protocol



Samba VFS: Functions by Category, todo

VFS Function Categories	Number	Todo
Path based	21	Use `O_PATH` handles
Path based namespace changing (create, delete, . . .)	8	Use `*at()` calls
Handle based but not allowed on `O_PATH` fds	8	Use /proc/PID/fd/FD
Handle based	42	-
DFS-related	3	-
Disk operations	9	-
Pure path to path translation	4	-
Special cases (eg FileIDs)	6	-
Sum todo	29	

Table 1: VFS interface functions by category needing changes

Challenges: Permissions and Oplocks

Opening a file handle requires at least `O_RDONLY`

- Path based `stat("dir/file")`
  `"x"` access right on `"dir"` required
- To replace `stat()` with `fstat()` first we to open the file
  1. `fd = open("file", O_RDONLY)`
  2. `fstat(fd)`

  `"r"` access right on `"file"` required

Kernel oplocks

- `O_RDONLY` triggers a kernel oplock break

O_PATH to the rescue

O_PATH

The solution: Linux open() flag O_PATH

- Available since since Linux 2.6.39 (May 2011), in FreeBSD 14
- Returns a file handle that acts as a mere path "reference"
  - I coined the term pathref for referring to them in Samba
- Doesn't need "r" on object, only "x" on the parent directory

Limitted number of syscalls are allowed

- `fstat(fd, ...)`, `fchdir(fd, ...)`, `utimensat(fd, ..., AT_EMPTY_PATH)`
- Syscalls that work at the file-descriptor/inode level
- Can't be used for any sort of IO
- Can also be used as dirfd for `*at()` syscalls

Fallback to open-as-root if O_PATH is not available

- root-opened fds are "guarded", access only via accessor functions
  - `fsp_get_pathref_fd(fsp)`, `fsp_get_io_fd(fsp)`
  - `fsp_get_pathref_fd(fsp)` must be auditted

O_PATH Cont.

Samba needs more then fstat()

- Samba needs to read ACLs and xattrs
- But both can't be retrieved via O_PATH handles
- Use the /proc/self/fd/FD trick:
  - use path based version with path "/proc/self/fd/%d"
  - replacing %d with the O_PATH fd

Example Code: Fallback to getxattr

```
if (fsp->fsp_flags.is_pathref) {
        char buf[PATH_MAX];
        sprintf(buf, "/proc/self/fd/%d", fd);
        getxattr(buf, ...);
} else {
        fgetxattr(fd, ...);
}
```

Fine Print

- /proc/self/fd currently Linux only, elsewhere fallback to path based access
- Which is the same net result as in pre O_PATH Samba

Challenge: SMB1 Heritage

Due to paths being used heavily in the protocol we have pervasive use of paths in the Samba codebase

- we want to convert 21 path based VFS functions, ...
- that are used at several hundred places in the codebase and ...
- will we need a file handle in all those places

Samba high-level code "degrades" handles to path-based access in many places

- So in theory we have a handle (`fsp` in Samba parlance)
- But use path attached to `fsp` (`fsp->fsp_name`) with path based VFS function
- Or need to call a VFS function on the parent directory of `fsp->fsp_name`
- Sometimes paths get passed to functions, not a handle – even though we have one

SMB1 Heritage: Turning pathnames into pathrefs

How to get a file handle? The old way

Samba's internal file handle structure is of type `struct files_struct` and all variable pointing to objects of such type are typically called `fsp`'s.

- `fsp`'s are returned by `SMB_VFS_CREATE_FILE()`
- this is the 1000 pounds Gorilla of the VFS functions zoo
- calls on to `SMB_VFS_OPENAT()` to open the low-level fd
- then goes through Samba's NTFS Windows emulation (eg `locking.tdb`)

New, additional way to get a `O_PATH` file handle

New helper function `openat_pathref_fsp()`:

- skips the NTFS emulation logic
- just calls `SMB_VFS_OPENAT()` with `O_PATH`
- returns a pathref fsp
- pathref fsps can be upgraded to "full" fsps
  - fd is reopened via `/proc/PID/fd/FD`
  - NTFS Windows emulation code is run

SMB1 Heritage, Cont.

Client supplied paths are processed by the core function `filename_convert()`

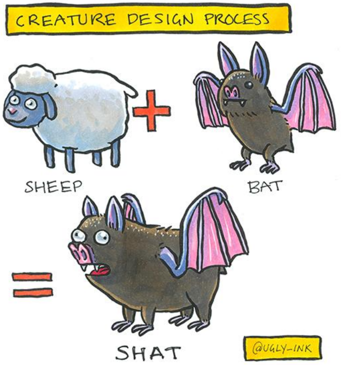
- Returs a pointer to an object of type `struct smb_filename`.
    - Variables are typically called `smb_fname`.
- `filename_convert()` is updated to call `openat_pathref_fsp()`
- storing the resulting pathref `fsp` inside `struct smb_filename`
    - `smb_fname->fsp`
- As a result the whole codebase has immediate access to a file handle.

This allowed converting the large codebase to a handle based VFS in a piecemeal fashion.

The Design Squad

The Design Squad

Stefan Metzmacher
Volker Lendecke
Jeremy Allison
Ralph Böhme



The Construction Squad

Construction Squad

Noel Power
Samuel Cabrero
Jeremy Allison
Ralph Böhme

The End

Thank you!

Ralph Böhme, SerNet Samba Team
slow@samba.org
rb@sernet.de

Links

- https://wiki.samba.org/index.php/The_New_VFS