



CIFS Unix/POSIX Extensions

An update ...

Steve French

Filesystem Architect - IBM LTC
Samba team
Linux CIFS maintainer ...





Legal Statement

This work represents the views of the author and does not necessarily reflect the views of IBM Corporation.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries: IBM (logo), A full list of U.S. trademarks owned by IBM may be found at <http://www.ibm.com/legal/copytrade.shtml>.

Linux is a registered trademark of Linus Torvalds.

Other company, product, and service names may be trademarks or service marks of others.



Who Am I?

- Author and maintainer of Linux cifs network file system, one of larger Linux file systems
- Architect for File Systems/NFS/Samba in IBM LTC
- Design/Developed various network file systems since 1989
- Member of the Samba team, coauthor of CIFS Technical Reference and former SNIA CIFS Working Group chair



Outline

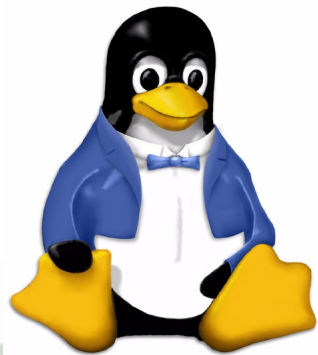
- SMB/CIFS Lives ... A short history
- New developments:
 - Unix/Linux Extensions continue to improve
 - SMB2
- Unix Extensions
 - Protocol status
 - Problematic file system operations
- Next Steps



Rebirth of SMB?



- Unix Extensions continue to be improved and implemented in various clients / servers
- Release of Vista (early 2007) included new default Network File System protocol: **SMB2**
- Prototype Implementations of SMB2 in Samba 4 by late 2006 (and Wireshark)



History of SMB/CIFS

- Birth of SMB/CIFS:
Dr. Barry Feigenbaum
et al of IBM
(published 1984 IBM
PC Conf), continued
by Intel, 3Com,
Microsoft and others



Became the default
for DOS, Windows,
OS/2, NT and various
other OS.

Evolved through
various “dialects”

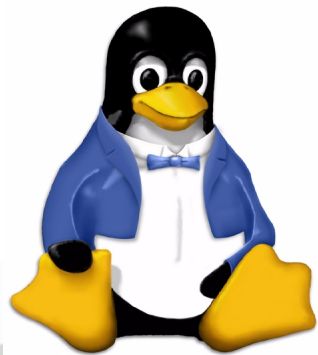
Happy 23rd Birthday!



New POSIX Extensions

- Share Encryption
- Proxy Capability
- Very large reads/writes





Features under Discussion

- Parallel CIFS
- Directory Caching
- Alternative transports
- API for common tasks
- Common “standard” mount options
 - make automount easier when mixed Unix/Linux clients

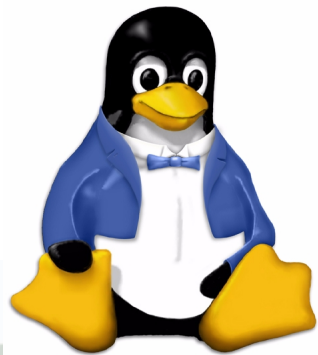


SMB2 Under the hood



- Not the same as CIFS but ... still reminiscent of SMB/CIFS
 - Same TCP port (445)
 - Small number of commands (all new) but similar underlying infolevels
 - Similar semantics






SMB2 vs. SMB/CIFS

- Header better aligned and expanded to 64 bytes (bigger uids, tids, pids)
- 0xFF “SMB” -> 0xFE “SMB”
- Very “open handle oriented” - all path based operations are gone (except OpenCreate)
- Redundant/Obsolete commands gone
- Bigger limits (e.g. File handle 64 bits)
- Better symlink support
- Improved DFS support
- “Durable File Handles”





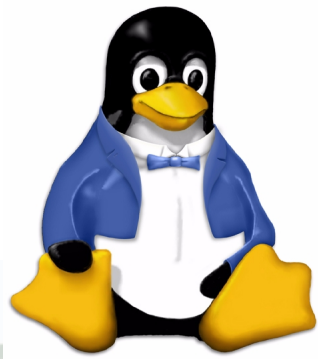
Adding Unix Extensions to SMB2

- SMB2 capability negotiation
 - SMB2_GLOBAL_caps returned on Negotiation
 - Sent on SessionSetup
- 



Other protocols

- SMB/CIFS has more than 80 distinct SMB commands (Linux CIFS client only needs to use 21). A few GetInfo/SetInfo calls, similar to SMB2, have multiple levels
- NFS version 2 had 17 commands (NFS version 3 added 8 more), but that does not count locking and mount which are outside protocol
- NFS version 4 has 37 commands (dropped some, added 25 more) but moved locking into core

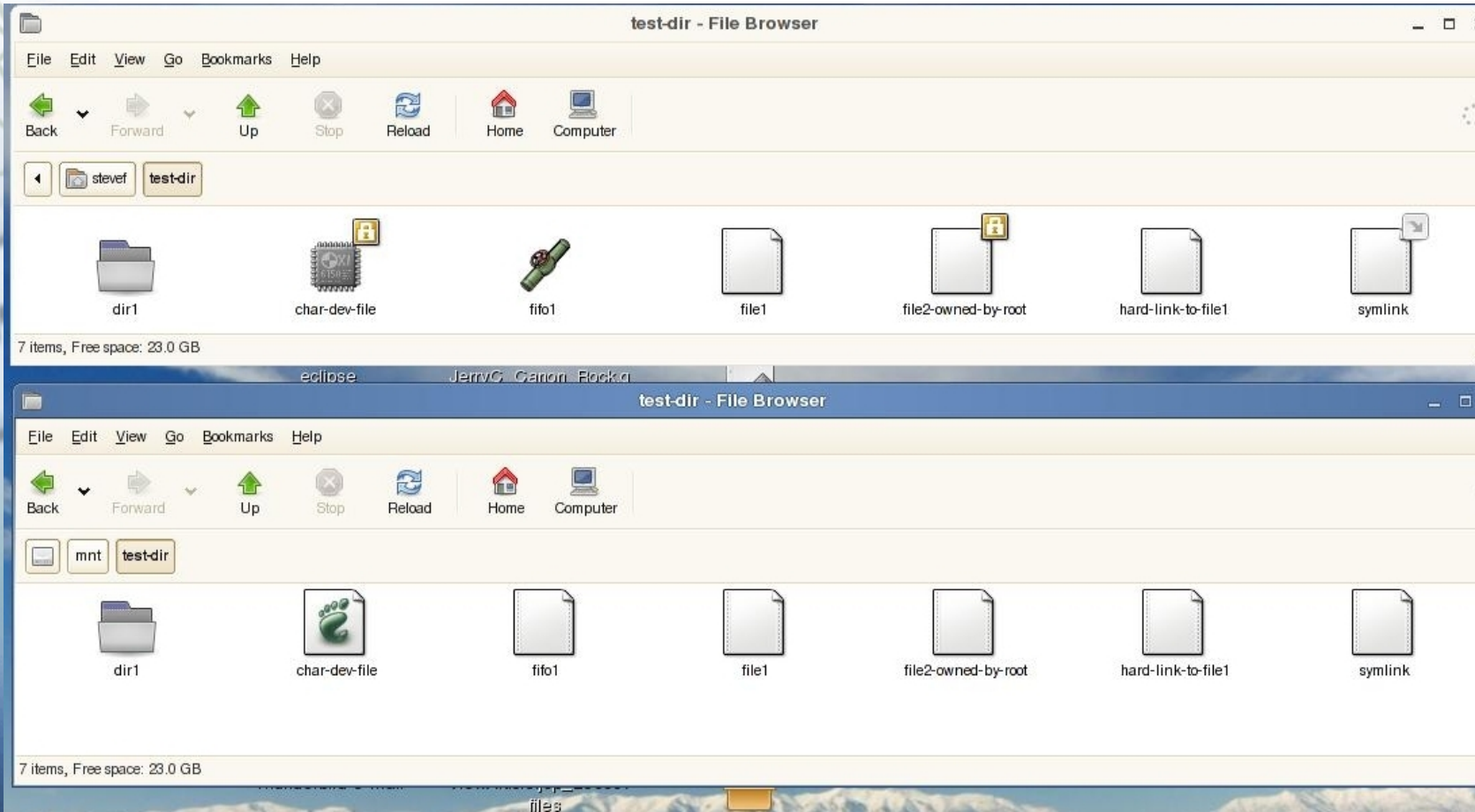


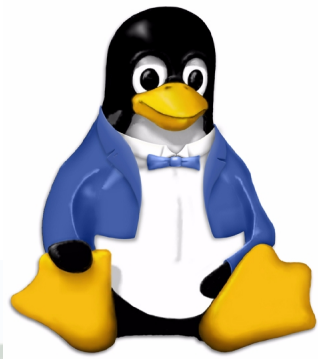
CIFS Linux (POSIX) Protocol Extensions

- The CIFS protocol without extensions requires awkward compensations to handle Linux
- Original CIFS Unix Extension (documented by HP for SNIA five years ago) was nice 1st step:
 - Required only modest extensions to server
 - Solved key problems for POSIX clients including:
 - How to return: UID/GID, mode
 - How to handle symlinks
 - How to handle special files (devices/fifos)

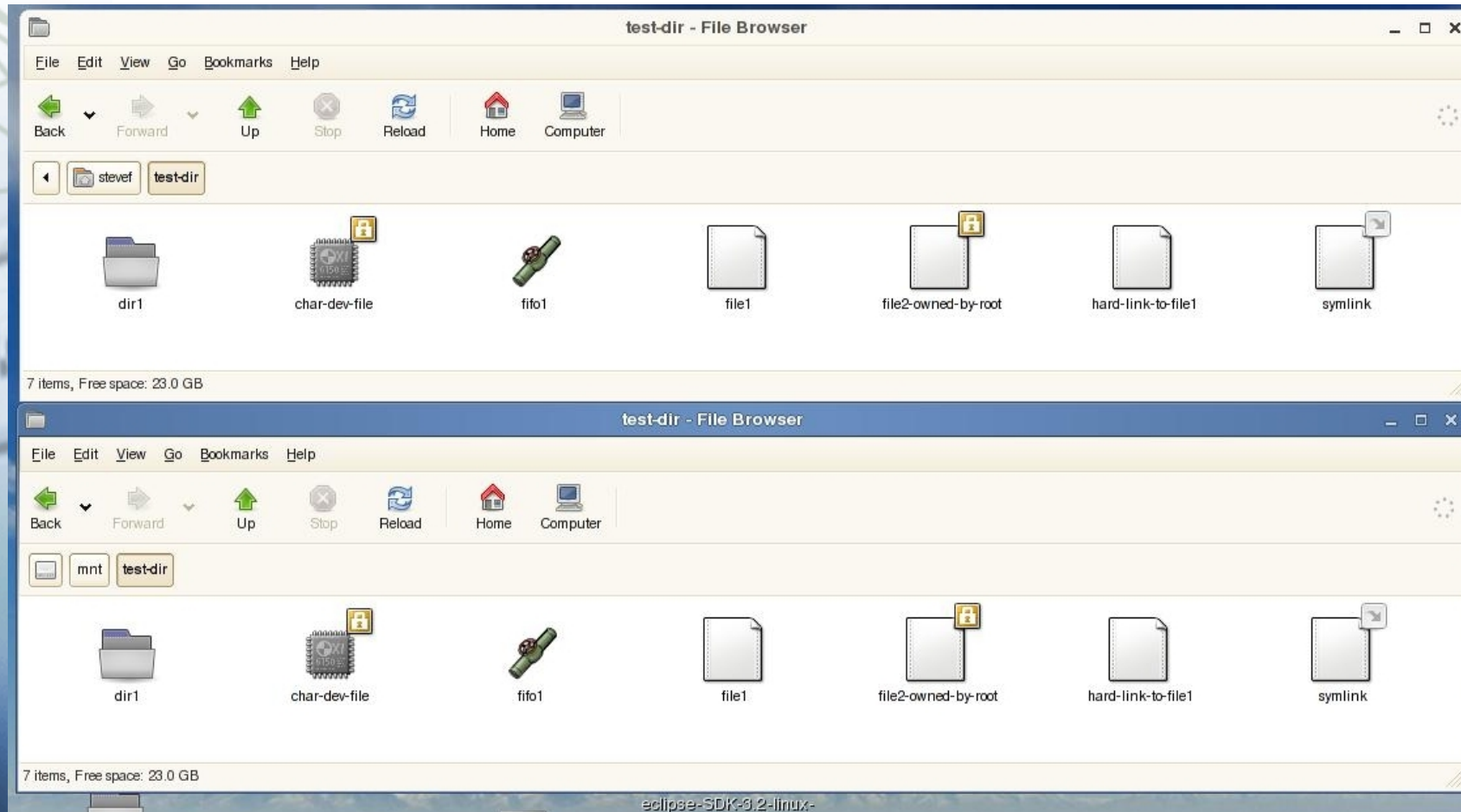


POSIX Conformance hard for original CIFS





CIFS with Protocol Extensions (CIFS Unix Extensions)



What about SFU approach?

- Lessons from SFU:
 - Map mode, group and user (SID) owner fields to ACLs
 - Do hardlinks via NT Rename
 - Get inode numbers
 - Remap illegal characters to Unicode reserved range
 - FIFOs and device files via OS/2 EAs on system files
- OK, **but not good enough** ...
 - Some POSIX byte range lock tests fail
 - Semantics are awkward for symlinks, devices
 - UID mapping a mess
 - Performance slow
 - Operations less atomic and not robust enough
 - Rename/delete semantics hard to make reliable



Original CIFS Unix Extensions

- Problem ... a lot was missing:
 - ▶ Way to negotiate per mount capabilities
 - ▶ POSIX byte range locking
 - ▶ ACL alternative (such as POSIX ACLs)
 - ▶ A way to handle some key fields in statfs
 - ▶ Way to handle various newer vfs entry points
 - lsattr/chattr
 - Inotify
 - New xattr (EA) namespaces



Original Unix Extensions Missing POSIX ACLs and statfs info

```
smf-t41p:/home/stevef # getfacl /mnt/test-dir/file1
# file: mnt/test-dir/file1
# owner: root
# group: root
user::rwx
group::rw-
other::rwx
```

```
smf-t41p:/home/stevef # stat -f /mnt1
File: "/mnt1"
  ID: 0          Namelen: 4096      Type: UNKNOWN (0xff534d42)
Block size: 1024      Fundamental block size: 1024
Blocks: Total: 521748      Free: 421028      Available: 421028
Inodes: Total: 0          Free: 0
```



With CIFS POSIX Extensions, ACLs and statfs better

```
smf-t41p:/home/stevef # getfacl /mnt/test-dir/file1
# file: mnt/test-dir/file1
# owner: stevef
# group: users
user::rw-
user:stevef:r--
group::r--
mask::r--
other::r--
```

```
smf-t41p:/home/stevef # stat -f /mnt1
  File: "/mnt1"
    ID: 0          Namelen: 4096      Type: UNKNOWN (0xff534d42)
Block size: 4096      Fundamental block size: 4096
Blocks: Total: 130437      Free: 111883      Available: 105257
Inodes: Total: 66400      Free: 66299
```



POSIX Locking

- Locking semantics differ between CIFS and POSIX at the application layer.
 - ▶ CIFS locking is mandatory, POSIX advisory.
 - ▶ CIFS locking stacks and is offset/length specific, POSIX locking merges and splits and the offset/lengths don't have to match.
 - ▶ CIFS locking is unsigned and absolute, POSIX locking is signed and relative.
 - ▶ POSIX close destroys all locks.



Protocol changes

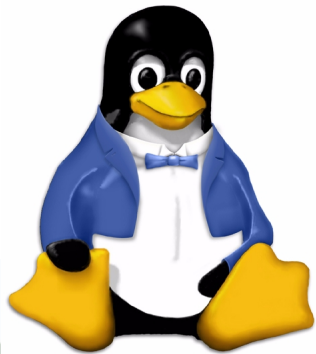
- The mandatory/advisory difference in locking semantics has an unexpected effect.
- READX/WRITE semantics must change when POSIX locks are negotiated.
 - ▶ Once POSIX locks are negotiated by the SETFSINFO call, the semantics of READ/WRITE CIFS calls change - they ignore existing read/write locks.
 - ▶ POSIX-extensions aware clients probably *want* these semantics.
 - It's a side effect, but a good one !



Windows client/POSIX interaction

- POSIX clients read/write requests conflict with Windows locks, but not POSIX locks (Windows locks are mandatory for POSIX clients).
- Windows clients read/write requests conflict with both Windows and POSIX locks (both lock types are mandatory for Windows clients).
- Windows locks are set, unlocked and canceled via LOCKINGX (0x24) call.
- POSIX locks are set and unlocked via the Trans2 SETFILEINFO call, and canceled via the NTCANCEL call.





Problematic Operations

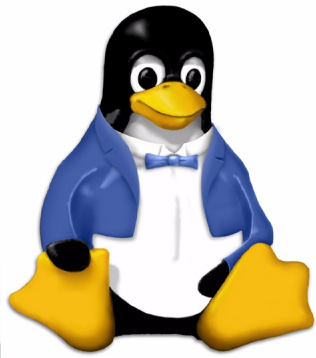




NFS not perfect ...

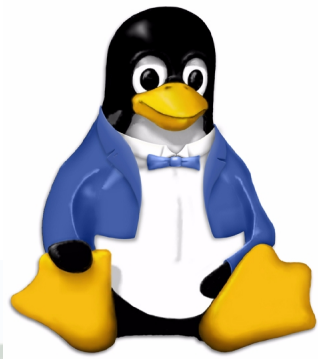
- Some are hard to address (NFS over TCP still can run into retransmission checksum issues <http://citeseer.ist.psu.edu/stone00when.html>)
- Silly rename sideeffects
- Byte Range Lock security
- Write semantics
- Lack of open operation lead to weak cache consistency model
- Most of these issues were addressed with NFSv4 as Mike Eisler pointed out (but NFSv4 has problems too)

CIFS has problems too



- There is an equivalent of “commit” but it is not as commonly used (ie to force server to flush its server side caches and write to metal)
- No grace period for lock/open recovery after server is rebooted (clients can race to reestablish state)





What makes network file system developers lives miserable?

- Constraints from network file system protocol
- Bugs in various servers that must be worked around
- Races with other clients
- Recovery after failure
- Long, unpredictable network latency
- Hostile internet (security)
- More complex deadlocks



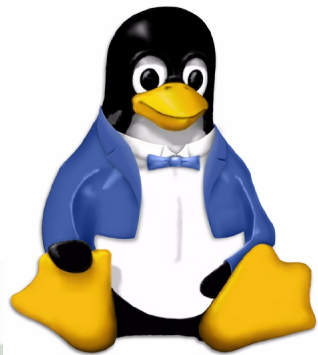
Don't (always) blame the protocol ...



- Some problems are with the implementation (e.g. nfs.ko, cifs.ko) not with the protocol
- It takes a long time to get implementations right ... current Linux ones are still tiny (under 30KLOC)



Beyond POSIX ... Linux Affinity Scorecard

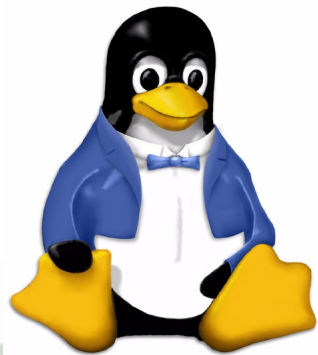


- Xattrs:
 - CIFS: Yes “user.” category only; NFS: no
- POSIX ACLs:
 - CIFS: Yes (w/Unix Extensions e.g. Samba, but mapping code to support Windows server not complete yet).
 - NFSv4: No NFSv3:Yes (Linux servers only)
- getlease/setlease fcntl
 - Neither CIFS nor NFS clients handle (CIFS protocol and servers would allow it though)
- lsattr/chflags
 - CIFS: yes (not to all servers) NFS: No
- DNOTIFY fcntl (or inotify)
 - NFS: No CIFS: No (but protocol & servers would allow)
- O_DIRECT NFS: yes; CIFS: No (but has as mount option for)



NFSv4 or CIFS for Unix?

- NFSv4 client in short term better performing in most (not all) workloads. Harder to configure for security though (AD is everywhere)
- With the newer Linux Extensions, CIFS to Samba is a great alternative and supports various Linux operations that NFS does not support
- CIFS (the implementation) missing some key features to catch up with competition
- CIFS will still be necessary for newer Windows until SMB2 support in kernel matures (we need to start now). To newer Windows servers use of SMB2 would be slightly better than CIFS
- Need to evaluate adding the Linux/Unix/POSIX extensions to SMB2 for Samba as we did with CIFS



Acknowledgements

Thanks to the Samba team, members of the SNIA CIFS technical work group, and others in analyzing and documenting the SMB/CIFS protocol and related protocols so well over the years. This is no easy task. In addition, thanks to the Wireshark team and Tridge for helping the world understand the SMB2 protocol better. Thanks to Jeremy Allison for helping me drive better Linux extensions for CIFS. Thanks to the Linux NFSv4 developers, the NFS RFC authors, and to Olaf Kirch, Mike Eisler for making less opaque the very complex NFSv4 protocol.

Thank you for your time!



More general improvements still needed in our aging protocol

- These changes were not really Unix or Linux specific but POSIX apps may have stricter assumptions
- Full local/remote transparency desired
- Need near perfect POSIX semantics over cifs
- Newer requirements
 - ▶ Better caching of directory information
 - ▶ Improved DFS (distributed name space)
 - ▶ Better Performance
 - ▶ Better recovery after network failure
 - ▶ QoS



Where to go from here?

- Discussions on samba-technical and linux-cifs-client mailing lists
- For Linux CIFS Extensions and CIFS: Wire layout is visible in `fs/cifs/cifspdu.h`
- CIFS and SMB2 information on MSDN now
- For Open Source contact Tridge about PFIF
- For SMB2, see the Samba 4 source
- Working on updated draft reference document for these cifs protocol extensions
- See http://samba.org/samba/CIFS_POSIX_extensions.html

