A Tour through the CIFS Protocol Extensions and Linux CIFS client

Steve French

File System Architect – IBM Linux Technology Center

http://svn.samba.org/samba/ftp/cifs-cvs/SDC08-cifsext.pdf

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?

- Architect for File Systems, NFS, Samba in IBM LTC
- Author and maintainer of one of the larger file systems (cifs)
- Designed/Developed various network file systems since 1989
- Member of the Samba team, coauthor of CIFS Technical Reference and former SNIA CIFS Working Group chair
- Among top 5 Linux fs contributors

Outline

- Why SMB/CIFS/SMB2?
- Unix Extensions
  - Motivation – Why do we need extensions?
  - Details
  - Future
- SMB2
- Linux CIFS Implementation
  - Features
  - Key progress in past year
  - Future

Why SMB/CIFS (and now SMB2)

?

Why SMB/CIFS?

- Ubiquitous
  - Supported by almost all Windows systems
  - Samba ships with all major Linux distributions, many Unix, MacOS, and various appliances
  - Dozens of other operating systems and appliances have other implementations
- Proven and reasonably well understood
  - 24 years and counting ...
  - X/Open CAE spec, SNIA CIFS Spec, and now more detailed WSPP information on MSDN

And because the alternatives have problems too ...

- NFS v3 or v4
- AFS/DFS
- HTTP/WebDav
- Cluster Filesystem Protocols

Olaf's "Why NFS Sucks" Talk at OLS 2006

The Nightmare Filesystem

NFS problems

- Some are hard to address (NFS over TCP still can run into retransmission checksum issues http://citeseer.ist.psu.edu/stone00when.html)
- Silly rename sideffects
- 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 questions even with NFSv4 ...

- Does extra layer between NFS and TCP (SunRPC), still required in v4, get in way?
- Can RPSEC_GSS performance overhead be reduced enough?
- ACL mapping problems (NFSv4 ACLs are almost NTFS/CIFS ACLs but not quite). Management of ACLs from both sides (Windows or CIFS vs. NFSv4) could break. What about the ACL mask?
- UID -> username@domain mapping overhead
- “stable file handles” and even stable file system ids still a pain on many modern fs!

And more to analyze for NFSv4

- "Close to Open"and cache consistency
- utimes -> fsync (hurts performance)
- "COMMIT" and periodic write stalls
- What about "**Linux Affinity**?" How well does NFSv4 or CIFS map to the Linux VFS entries needed by applications (not just the minimal POSIX file calls)
  - Similar question for other operating systems could be asked

Network File System Comparison

- Due to Windows and low end NAS interop CIFS has largest number of target servers
- NFS is faster in some, but not all
- NFSv4 still not as widely deployed, NFS v4.1 still too early
- SMB2 is likely to become important (default on Vista and has detailed documentation)
- Will WSPP lead to broader adoption?

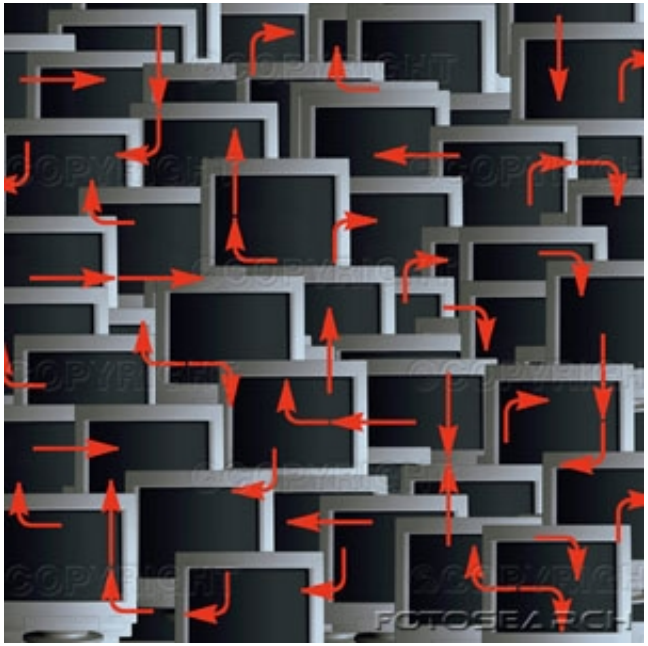
Cluster FS – No clear winner

- Network file systems are used far more often
- Too many cluster choices, no standard
  - OCFS2 – larger and more efficient
    - Lacks features needed for apps like Samba
  - GFS2 more full function, but not proven yet
  - Lustre vs. GPFS (neither in mainline Linux)
    - compete in high end clusters
    - IBM SOFS (Samba/NFS over GPFS/Linux cluster)
      - High performance and availability, transparent failover, active-active file exporters. Leverages ctdb
- NFS v4.1's optional pNFS feature may help

Don't always blame the protocol ..

- Some problems are with the implementation (e.g. s cifs.ko or mrxsmb10.sys) not with the protocol
- It takes a long time to get implementations right ... (e.g. current Linux one under 30KLOC)

WSPP will lead to wider adoption of SMB2/CIFS



- If:
  - Cluster & performance enhancements added
  - Unix interop addressed
  - Performance continues better than HTTP
- Especially in server room

From CIFS Unix Extensions to "POSIX Extensions"

The Quest for Perfect File Semantics

- We need “perfect” semantics on the wire
  - No difference between local and remote semantics (this is more than “POSIX file API”)
  - All file API expected to run on a local fs should be supported remotely too
- Unix Extensions fill in the holes needed for this ...

Some examples

- Unix uid/gid
- Statfs fields
- "POSIX ACLs"
- POSIX locking, posix write semantics
- Change notification (fcntl_dnotify and inotify)
- 64 bit increases to internal Unix structs
- Xattrs (Eas) – included trusted/security categories
- extended attribute flags (chflags/lsattr)
- More atomic file operations
- Async operations

Problem Unix file system ops

	Priority	Can do with CIFS	Linux client compensates
POSIX Mode	Very High	Fair	Yes (Experimental)
UID/GID	Very High	Fair	No
Hardlinks	Very High	Yes	Yes
Symlinks	High	Fair (Yes for SMB2)	Partial
statfs	Medium	Poor	Yes
special files (fifo, pipe)	Medium	Fair	Yes
POSIX acls	Lower	Fair	No
POSIX locks	High	Poor	Yes
lsattr/chflags	Lower	No	No
case sensitivity	High	Yes	Yes
time granularity	Lower	No	No
64 bit inode numbers	Lower	Varies	Yes
inotify	Medium	Fair	No
POSIX mv,rm semantics	High	Poor	Yes
trusted/security xattrs	Lower	Poor	No
POSIX open	Medium	No	No
POSIX pathnames	High	Good	Yes
“access” / “permission”	Medium	Poor (Yes SMB2)	Partial
exportfs (nfs srv) ops	Medium	Poor	Partial

How Unix Extensions addressed

	When implemented	Negotiated on tid	Which level
POSIX Mode	original Unix Extensions	No	SetInfo 0x200
UID/GID	original	No	SetInfo 0x200
Hardlinks	original	No	SetInfo 0x203
Symlinks	original	No	SetInfo 0x201
statfs	POSIX extensions	Yes	QFS 0x201
special files (fifo, pipe)	original, needed change	No	SetInfo 0x200
POSIX acls	POSIX extensions	Yes	SetInfo 0x204
POSIX locks	needed major change	Yes	SetInfo 0x208
lsattr/chflags	POSIX extensions	Yes	SetInfo 0x206
case sensitivity	original, needed change	Yes (on session)	SMB flag
time granularity	Future	Future	Future QFS
64 bit inode numbers	original	No	QueryInfo 0x200
inotify	Not possible, future	Future	Future SetInfo
POSIX mv,rm semantics	needed major change	Yes	Get with POSIX open
trusted/security xattrs	Future	Future	SetInfo 0x205
POSIX open/mkdir	POSIX extensions	Yes	SetInfo 0x209
POSIX pathnames	POSIX extensions	Yes	SetFSInfo flag
Very large read/write	POSIX extensions	Yes	READX/WRITEX (with Setfs on share)
Transport Encryption	POSIX extensions	Yes	QFS 0x203
Proxy	POSIX extensions	Yes	QFS 0x204
Who Am I	POSIX extensions	No	QFS 0x202

What is a File System?

- "a file system is a set of abstract data types that are implemented for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data" [http://en.wikipedia.org/wiki/Filesystem]
- A Linux kernel module used to access files and directories. A file system provides access to this data for applications and system programs through consistent, standard interfaces exported by the VFS
- This is much, much harder over a network … which is why making **Network File Systems** is fun
- An source of infinite bugs and features to keep software developers busy forever …

File System development is hard! What makes File Systems developers lives miserable?

- Intermittent problems that are incredibly hard to trace
- Huge amounts of data to sift through
- Flaky hardware
- Over time … exponential increase in disk errors

What makes network file system developers lives miserable?

- Constraints from network fs 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 and locking

All network fs can handle simple inode operations

- Linux inode operations
  - create
  - mkdir
  - unlink (delete)
  - rmdir
  - mknod
- Note vfs operations not all atomic (sometimes POSIX calls generate more than one vfs op although “lookup intents” help for nasty case of create) Some compensations are needed

•Linux has complex FS operations to implement

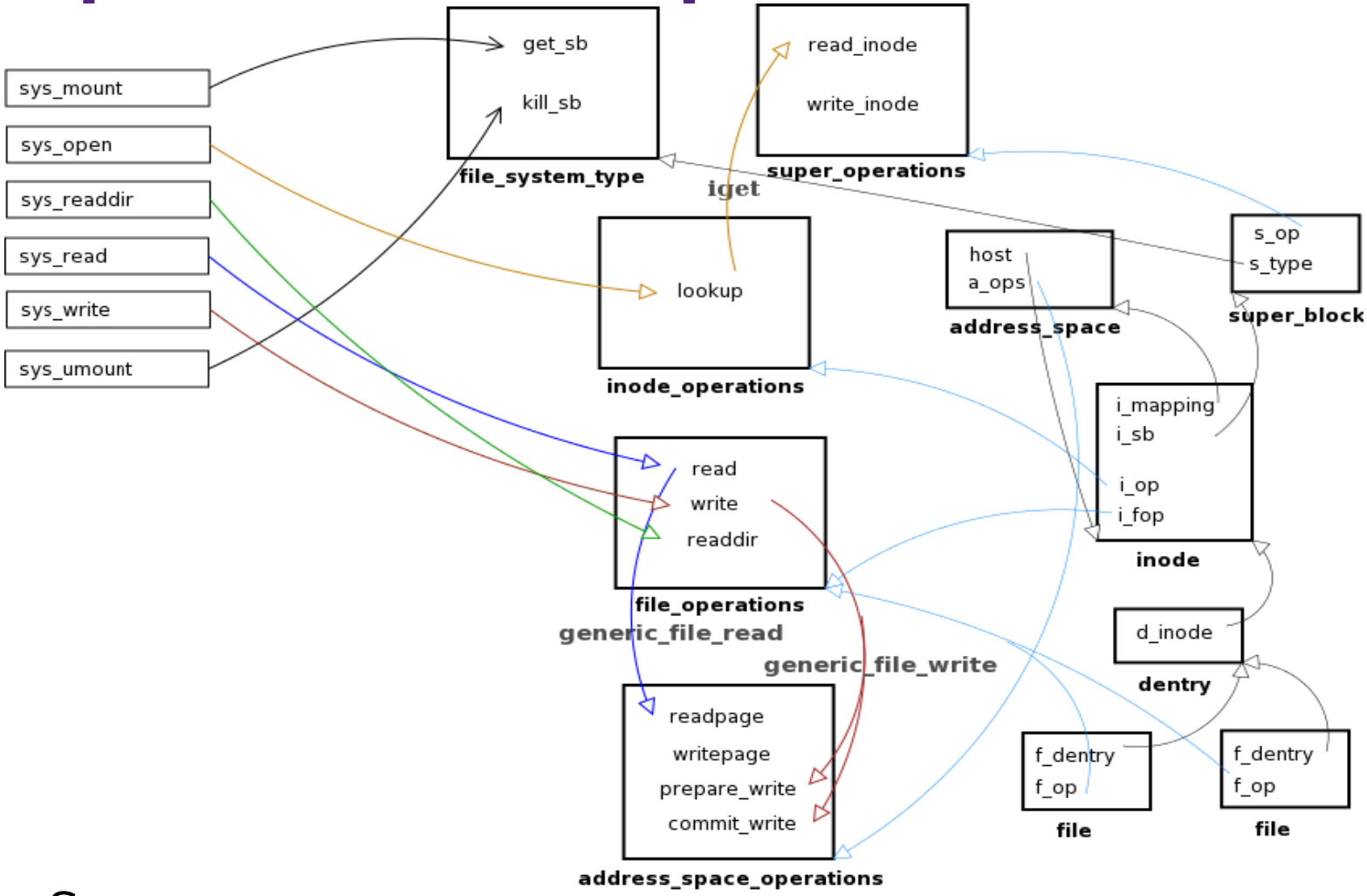


Fig: System call mapping and data structure relationships

Source: http://www.geocities.com/ravikiran_uvs/articles/rkfs.htm

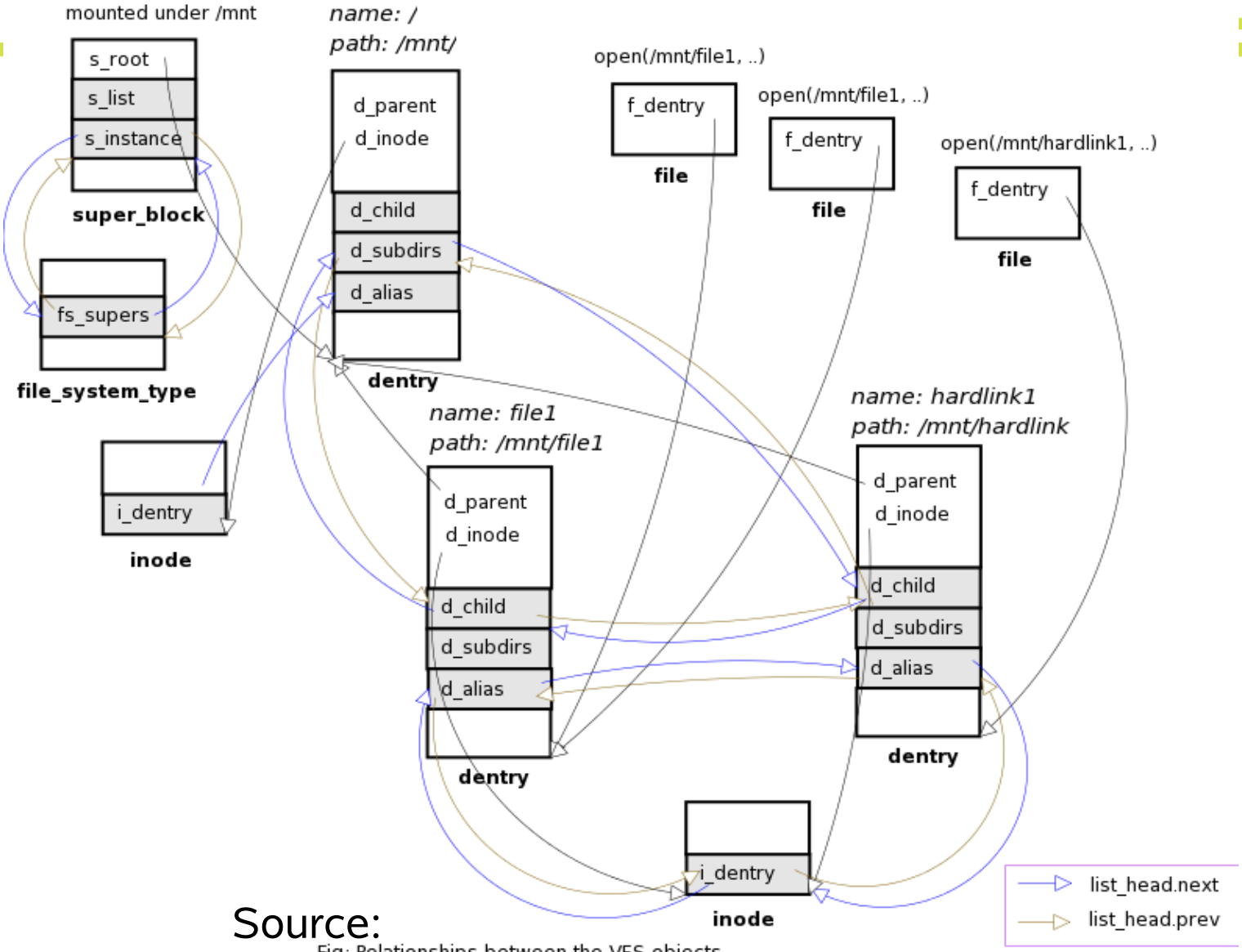


Fig: Relationships between the VFS objects

Source: http://www.geocities.com/ravikiran_uvs/articles/rkfs.htm

But we also want great performance

- We need great performance
  - Over GigE and faster networks
- And not crash when run over long latency networks
- Larger i/o sizes
- More efficient write/lock checks
- pCIFS?

Future

- Working with James Peach on more formal document defining current extensions (perhaps Draft RFC)
  - In the meantime see: http://wiki.samba.org/index.php/UNIX_Extensior (also see Linux cifs client fs/cifs/cifspdu.h)
- Ronnie Sahlberg and I (and others) have been discussing options for pCIFS and also DFS load balancing improvements

SMB2:

An Ancient Protocol Reborn

Back where we started!

- Ancient NFS and SMB born mid-1980s quickly popular
- Lots of other network file systems died out in between
- HTTP/WebDAV too slow, and can't do POSIX
- No widely deployed cluster fs standard
- 2008: Back where we started with NFSv4 and SMB2 widely deployed and going to be dominant?

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” - most path based operations are gone
- dynamic “credits” instead of max_mux
- Redundant/Obsolete commands gone
- Bigger limits (e.g. File handle 64 bits)
- Better symlink support
- Improved DFS support
- “Durable File Handles”

Unix Extensions to SMB2?

- Biggest gaps
  - Unix UID/GID, mode returned on lookup (getattr) and chmod (setattr)
  - Support for all fields in statfs (e.g. “df”)
  - Posix create/mkdir
  - How to indicate posix rather than ntfs semantics on handle based calls
  - Advisory byte range locking
- Also others: POSIX ACLs, chflags/lsattr

Unix Extensions to SMB2?

- Needed?
  - Yes! Must get UID/GID efficiently and accurately, posix file semantics, statfs ...
- Options (one or more of following)
  - Reserve new command codes
  - Negotiate new dialect
  - Send via new ioctls
  - Reuse a subset of existing infolevels for Transact2 Set/GetFileInfo
    - At first was my preference since changes least code

Another alternative

- Suggested by George Colley
  - Use “SMB2_CREATE_CONTEXT” struct
  - Specified in SMB2_CREATE command
  - Not mutually exclusive with others: Can append more than one context
  - Type of context indicated by 4 character name
  - Can be used for Get/Set fileinfo and and open
    - Could be superset (unix specific info + windows, or just unix info not returned by smb2 infolevels)
  - Requires us to reserve a context name
  - Does not address all needs (e.g. fsinfo)

Context continued

- Context Name:
  - "PSXQ" (posix file query)
  - "PSXS" (posix set file information)
  - Struct posix_file_request
    - Request Includes: flags to request support for specific features on this handle (posix paths, posix byte range locks, posix acls, posix file semantics, return posix file info) and mask which indicates which flags are known
    - Response returns which flags could be supported and mask indicating which flags are understood
    - Add version field?

Context continued

- Certain behaviors not possible with this approach
  - e.g. mixing posix and windows locks on same handles
  - Open -> query -> set -> close (have to do openquery->close->openset->close)
- Query/SetFS could be done by opening the root directory, but alternate mechanism may be preferable

Context can set/return various structs

- The context begins with:
  - Behavior flags (request posix pathnames, locks on this handle etc.)
  - Count of structures
- Can set/query one or more of following structs:
  - Query/SetFileInfo (returns Unix subset of FILE_UNIX_INFO)
  - Statfs (returns Unix subset of POSIX FS Info)
  - Query/SetPosixACL
  - Query/Set xattrs (trusted, security etc,)
  - Query/Set attr flags (lsattr/chflags)

Unix Extensions in SMB2

	Operates on	Implement via which "context level"
POSIX Mode	Handle	Via Create context, SetFileInfo
UID/GID	Handle	Via Create context, SetFileInfo
Hardlinks	Handle	In SMB2
Symlinks	Handle	In SMB2
statfs	tid	Via Create context, QueryFSInfo
special files (fifo, pipe)	Handle	Via Create context, SetFileInfo
POSIX acls	Handle	Via Create context, SetPosixACL
POSIX locks	Handle	Flag in Create context
lsattr/chflags	Handle	Via Create context, QueryFileInfo2
case sensitivity	Handle	SMB2 flag
time granularity	either	CreateContext, QueryFSInfo
64 bit inode numbers	Handle	In SMB2
inotify	Handle	Via Create context, SetInotifyInfo
POSIX mv,rm semantics	Handle	Flag in Create context
trusted/security xattrs	Handle	CreateContext, SetXattr
POSIX open/mkdir	Handle	Flag in Create context
POSIX pathnames	Handle	Flag in Create context
Very large read/write	Handle	Flag in Create context
Transport Encryption	tid	Via Create context, set encryption?
Proxy	tid	Ioctl?
Who Am I	session (or tid)	Via Create context, whoami

Linux CIFS:

A Year in Review

In the middle of exciting changes

- From 2.6.22 on July 8th, 2008
  - "Holy Dancing Manatees, Batman!"
- To 2.6.26 on July 13, 2008
  - "Rotary Wombat"

A year in review ...

- The Linux File System layer (VFS) grew about 7% in size
- The FS overall (VFS and individual file systems) grew about 6% from 487KLOC to 518KLOC
- CIFS grew as well. Even considering just the kernel portion:
  - Over 250 changesets from almost 50 different developers
  - 10K changed/new - now more than 24K lines of code

Highlights (CIFS)

- Ipv6 support (2.6.22)
- Additional POSIX extensions for improved app compatibility especially on mounts to Samba (2.6.22)
- Cifs acl support (optionally get/set POSIX mode via CIFS/NTFS ACLs)
- Kerberos Support (Improved secure, enterprise authentication)
- DFS (Global Name Space)

Highlights (Samba)

- WSPP PFIF agreement (!!)
- Samba clustering (!)
  - Tridge's Ctdb “clustered tiny data base API” is brilliant
  - Improved load balancing, availability
  - IBM offers SOFS (“Scale Out File Services”) highlighting Samba/NFS/GPFS and performance has been excellent. Tivoli Storage Manager integration also helpful
- LikewiseOpen released

Highlights (Samba) continued

- Now Can administer Linux much better from Windows (MMC) via Samba than before
- Samba performance improvements (including splice support)
- Server support for new posix extensions to cifs protocol
- “per-share encryption” feature
- SMB2 support (Samba 4 only)
- Better libraries for others to use

Samba 3

- Samba 3 security point release issued on May 28rd (3.0.30) and 3.0.31 released on July 10th, Maintainence release 3.0.32 on August 25th
  - All customers should migrate to at least 3.0.29
- Samba 3.2 released July 1st. 3.2.2 on August 19
- New POSIX Extensions
  - Share Encryption
  - Proxy Capability
  - Very large reads/writes

Samba 4

- Samba 4 alpha 4 released on June 5th
- Samba 4 not exactly “experimental release” since other packages (OpenChange e.g.) now leveraging Samba 4 client libraries etc.
  - Samba 4 does lack some key features Samba 3 has, and Samba 4 gets less distro testing). Samba 4 sub-team small
- Samba 4 testcases: gentest, smbtorture have been very useful in identifying holes in MS documentation, not just in improving server code quality
- Eventual Samba 3 / Samba 4 merge ?

Samba 4

- Samba server improving incredibly rapidly
  - In merged tree, now 1,518,748 LOC (measured using David Wheeler's sloccount):
    - samba4 directory is 851,556 LOC
    - samba3 directory is 605,709
- SMB2 server code now passes most of Microsoft's testcases already!
- Smaller changes than expected due to good design

- For further reading:
  - WSPP documentation (on msdn)
  - Samba web site
    - http://www.samba.org
  - CIFS Extensions Wiki
    - http://wiki.samba.org/index.php/UNIX_Extensions
  - CIFS Project web site
    - http://linux-cifs.samba.org

Thank You for your time!