Samba Internals

By Jeremy Allison

Development Team

email: jra@samba.org

Why document the internals of the Samba code ?

- Samba, in conjunction with Linux or FreeBSD is now used in many "appliance" file and print serving products.
- Samba is a part of every Linux distribution.
- Samba has now been adopted by major software vendors such as SGI, HP, and Veritas.
- The design philosophy behind Samba is not clear from looking at the code.
  - Why certain features may be accepted or rejected depends on how they fit into the overall design.

Quick Overview

- Samba consists of two user mode daemons.
  - nmbd - NetBIOS naming daemon. Not covered further in this talk.
  - smbd - Main file and print serving code.
- smbd has evolved over seven years of coding.
- Originally a file server, it has expanded to include print services, authentication services and now an implementation of an entire RPC protocol.
  - smbd is too complex. Much work is being done to simplify it and break it into managable parts.

smbd design

- smbd consists of a single process per connected "client".
  - Multi-user Windows servers such as Citrix or Terminal server can break this assumption.
- UNIX user context is used for security.
  - This is a very important point. smbd does not enforce security itself, it sets the effictive userid to the UNIX uid mapped to the client context and lets the OS determine access. No "root race" holes.
  - As a consequence of this smbd is single threaded. POSIX threads are not guaranteed to have a security context.

smbd event flow

- Dealing with incoming SMB messages [smbd/process.c]:
  - select()
  - read message from socket into 64k buffer [inbuf].
  - check message for correctness
  - check incoming user context, change effective uid if needed.
  - process message (done in switch_message()) - generating reply into 64k buffer [outbuf].
  - send reply to client.
  - periodically do general housekeeping (timout events etc.)

smbd design (continued)

- As close to Windows semantics as POSIX allows.
- Try to overlay POSIX filesystem with Windows semantics in the core code.
  - Don't create a "shadow" filesystem with dot files.
  - Don't create mappings that have no meaning to the underlying system (ACL or user databases).
  - No modifying file contents (no CR/LF translation).
- VFS layer in Samba 3.0 will provide "pluggable" mechanisms to provide this kind of OEM customization.

Mapping Win32 concepts to POSIX

- Win32 has some concepts that don't map well to POSIX.
  - Deny modes.
  - ChangeNotify.
  - NT specific ACL's.
  - Oplocks
  - Differences in byte-range locks.
- Samba implements deny modes between smbd processes via a shared memory area.
  - POSIX processes ignore the deny modes.

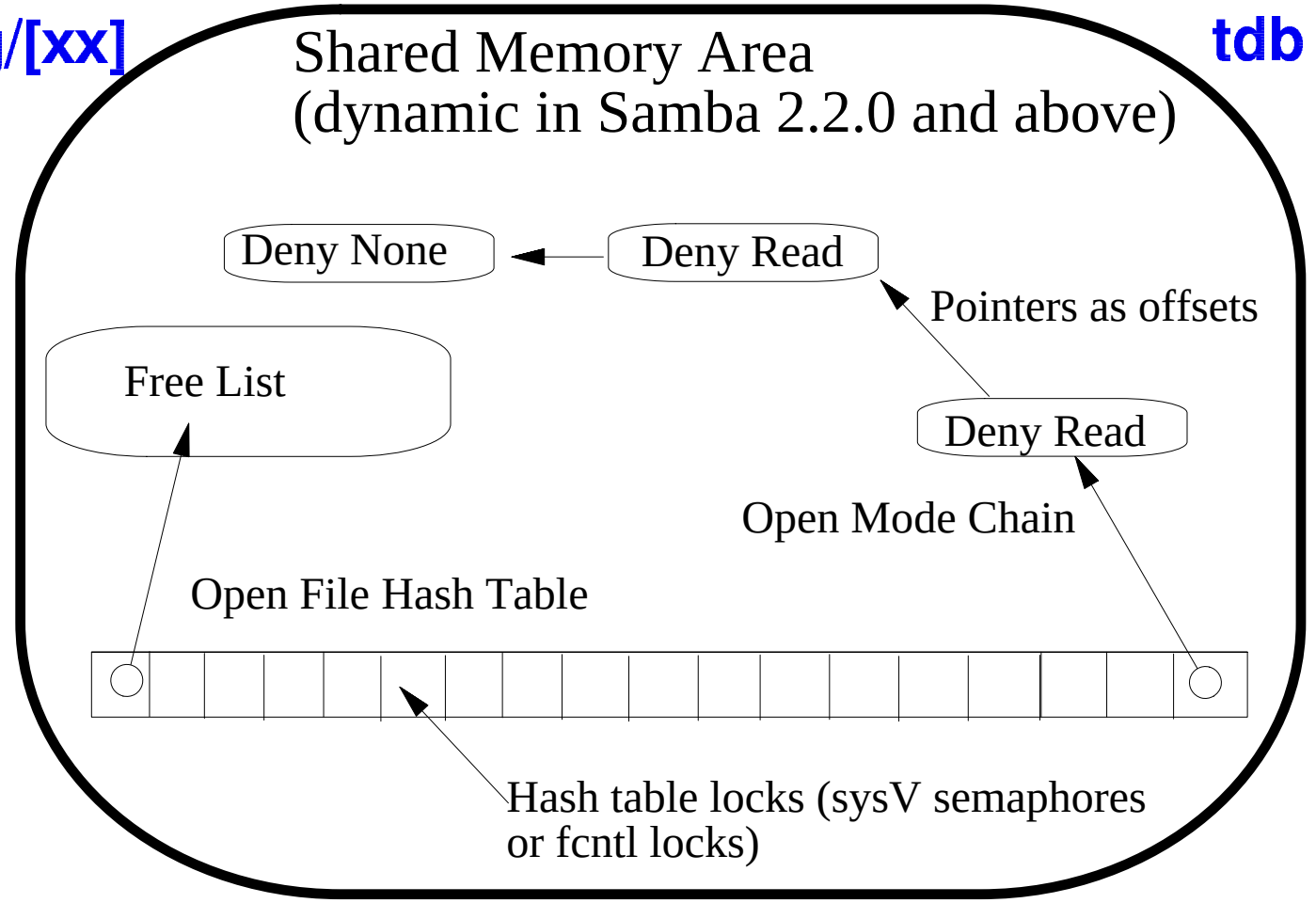
Deny mode semantics in POSIX

- POSIX has no "deny modes". Samba layers these over ordinary POSIX open calls [smbd/open.c].
  - POSIX apps do not interact with DENY modes.
  - Reason - what happens if someone opens /etc/passwd with DENY_ALL ?
  - DENY mode semantics are not logical - adding this to POSIX is not good design.
- Samba implements a fast, smbd to smbd mechanism to convey deny modes between user processes.
  - No centralized deny mode daemon needed.

Samba shared memory Deny mode database

locking/[xx] in 2.0.x

tdb in 2.2.0



Creating Oplocks in POSIX

- Allowing Oplocks on top of POSIX breaks consistant view of filesystem (and Samba philosophy) [smbd/oplocks.c]
  - However, too useful not to implment. Needed for SMB speed.
- Deny mode database holds all shared info about open file state. Oplock records added to this data.
- Blocking IPC mechanism between smbds needed that would integrate into select()/poll().
- UDP messages on loopback interface chosen.

Oplock communications

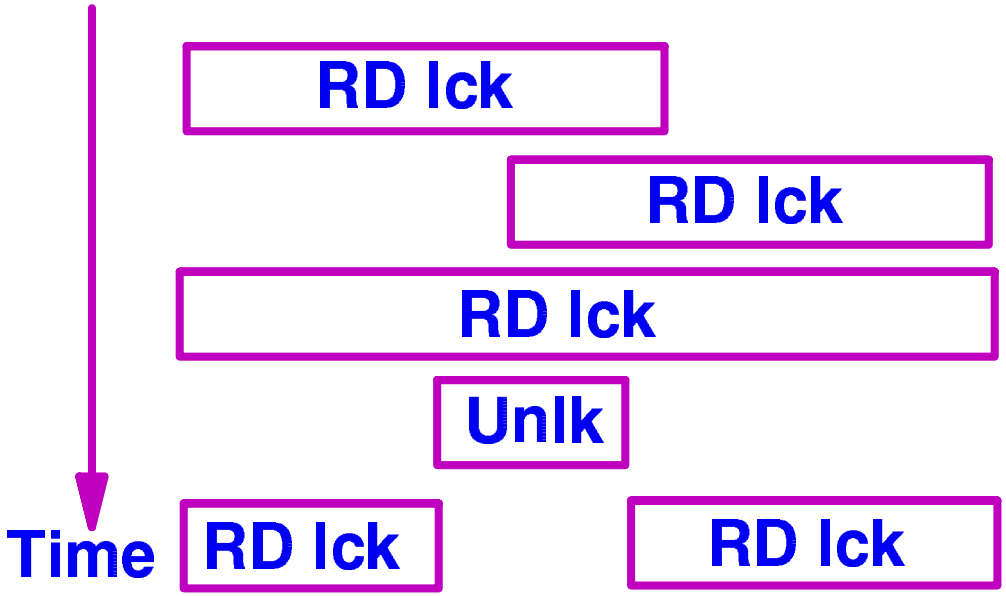
- On break request, smbd locks db, finds holder of oplock, sends break request via UDP port, releases db lock then blocks awaiting reply).
  - Code in [smbd/open.c] and [smd/oplock.c] - request_oplock_break() function.
- Receiver smbd gives priority to incoming UDP messages in select(), recurses into secondary smbd processing loop [smbd/oplock.c].
  - "Dangerous" messages that may cause an oplock break from the receiving smbd are queued at this time.
- On exit from recursed state, queued messages are given priority [smbd/process.c] - receive_message_or_smb().

The swamp - mapping Win32 byte range locks to POSIX

- Win32 byte range locks seem to be easy to map into POSIX.
  - Approach chosed in all Samba versions 2.0.x and before.
  - Depends upon locking conflicts being handled at client redirector.
  - Not possible to give exact Windows semantics.
- Samba 2.2.x and 3.0 have correct Win32 semantics.
  - "Correct" here means 'what NT does'. Has little relation to Win32 documentation or the spec.

POSIX locks - the exact semantics

- Lock ranges can be merged/split.
- Lock ranges can be upgraded/downgraded.
- 32/64 bit signed, not unsigned ranges.



Kernel view of locks

POSIX lock semantics (continued).

- Killer issue : POSIX locks are **per process**, not per file descriptor.
- Eg:

```
int fd1 = open("/tmp/bibble", O_RDWR);
fcntl(fd1, F_SETLK, &lock_struct);
fd2 = dup(fd1);
close(fd2);
```

SUPRISE ! The lock you thought you had on fd1 is now gone !

In anyones wildest dreams this is not desirable behaviour.

POSIX lock semantics (continued).

- Samba 2.0.x solution to this problem was to reference count all opens on a file onto a single fd, open read/write (if possible).
  - Conserves fd useage.
  - Samba checks prohibited security overrides.
- Disadvantages are :
  - Multiple opens under different uids - need to use fork() as a proceedure call to check return.
  - smbd is lying to operating system about access mode.
- 2.2.0/3.0 solution - store pending closes in a tdb.
  - Allows multiple opens to obey Samba philosophy.

Mapping Win32 concepts to POSIX (continued)

- ChangeNotify is a problem as it is resource intensive.
  - Similar to FAM on IRIX.
  - For portability reasons, Samba currently does a periodic scan, with no depth.
  - FAM-style interface is planned. User feedback needed.
- NT ACL's.
  - Current plans are to map to simple UNIX permissions as a first step.
  - More NT RPC calls needed to allow users to see IRIX users as NT owners.

Mapping Win32 concepts to POSIX (continued)

- Byte-range locks have different semantics under Win32.
  - No range splitting or merging.
  - No lock upgrades/downgrades.
  - Few apps depend on these exact semantics.
  - Win32 redirector helps here by filtering invalid application requests on the client.
- Blocking locks are the only real issue.
  - Samba is not multi-threaded so implements this by periodic polling.
  - Future work may add some threading to Samba for limited purposes. Full multi-threading not planned.

Resources

- Samba for IRIX Web site :
  - http://www.sgi.com/software/samba
- Main Samba Web site :
  - http://samba.org
- Newsgroup :
  - news:comp.protocols.smb
- Samba discussion list :
  - email: samba@samba.org
- Samba development list :
  - email: samba-technical@samba.org