SMB3 POSIX Extensions

Client Perspective and Server Perspective

Jeremy Allison
Steve French

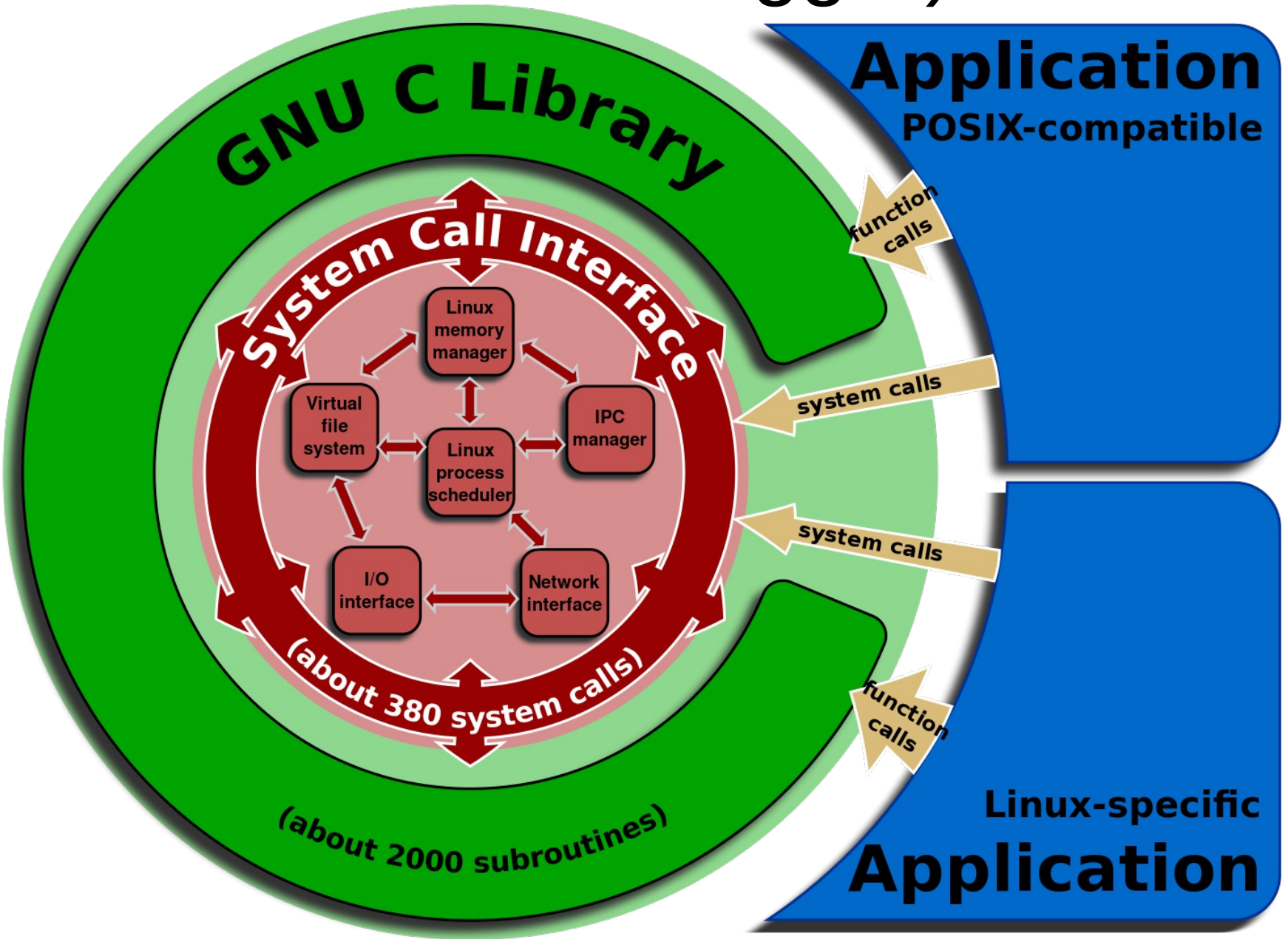
Legal Statement

- This work represents the views of the author(s) and does not necessarily reflect the views of Primary Data Corporation or Google
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.

Outline

- What is POSIX?
- Why do these extensions matter?
- What if we don't have them?
  - What works?
  - Some history: CIFS Extensions
  - Alternatives
- Client Perspective
- Server Perspective

- POSIX != Linux (they are close but Linux API is bigger)



Motivations for Extensions

- Linux Apps work!
  - Case sensitivity e.g. is required for the kernel to build on Linux
  - (And other posix-like operating systems who want posix behavior for files)
- Improve common situations where customers have Linux and Windows and Mac clients accessing the same data
- Deprecation of CIFS – make sure extensions work with most secure, most optimal SMB3.1.1 dialect

What works

- Without Extensions
  - Demo

Other Alternatives: AAPL

CIFS Unix/POSIX Extensions

- What was wrong with what we had?
  - Remember CIFS Deprecation?
  - And not just due to WannaCry …
    - SMB3 is really good ...

Client Perspective

- What about the Linux Kernel?
  - What does it really need from SMB3 to be optimal…?
  - Not just ability to do 'cool' things like compile the kernel on SMB3 mount, boot linux (and show blazing performance …!)
  - For all key features: SMB3 >= CIFS with/Unix Extensions
    - We are not asking user to go backwards

The challenges of Create/Rename/Delete

The challenges of POSIX inode metadata

- What do we need to be able to return?
- What about mode bits and ACLs?

The Challenges of POSIX locking

The Challenges of POSIX FS info

Server Perspective

- What does a server need the client to send? What protocol optimizations does the server need to make POSIX extensions usable?

Server Perspective

- Learn from the mistakes of SMB1 Unix extensions.
  - Security issues paramount.
  - Remove the possibility of server-followed symlinks
    - Break interoperability with NFS :-(, but necessary.
- Minimum Necessary Change (with apologies to Asimov's "*The End of Eternity*").
  - Fewer changes to the protocol the better.
  - Use the fact that we have experience with Samba in sharing between Windows and UNIX SMB connections.

Server Perspective Continued..

- Server-followed symlinks that the client can create have been a security disaster in Samba.
- Server-following symlinks is a useful holdover from ancient times, when admin-created symlinks gave great flexibility to setups.
  - As soon as clients gained the ability via UNIX extensions to create symlinks, disaster strikes.
  - Failed design decision to store these as real symlinks on the server filesystem.
    - Convenience for dual NFS / SMB1 servers.
- **THIS MUST NOT BE ALLOWED FOR SMB2+**

Server Perspective Continued..

- The key for SMB2 UNIX extensions is to allow simultaneous Windows and UNIX handles – using SMB2 create contexts.
  - Adding UNIX extension create context turns on POSIX behavior for this handle only.
  - Allows client code to probe for POSIX behavior – SMB2 specifies unknown create contexts are ignored.
  - The Samba server already has to handle this case in serving POSIX and non-POSIX client simultaneously.
- Leads to new Negotiate context requirement from the server.
  - That way a client can determine if a server could support POSIX behavior on a handle, but choses not to.
  - POSIX servers may expose POSIX behaviors or deny them depending on pathname (crossing mount points).

Server Perspective Continued..

- The rest of the changes are relatively small.
- One new info level needed to cope with POSIX stat returns.
- Keep protocol as close to “native” Windows as possible.
  - Map POSIX ‘mode’ into Windows ACL encoding.
  - No POSIX ACLs – return everything as Windows ACLs.
  - No POSIX uid/gids – return everything as Windows SIDs.
    - Client systems must cope with mapping SIDs anyway.
- Filename handling (POSIX specific, case sensitive) is the largest change. No access to Windows streams.
  - If you want a Windows stream handle, open a Windows stream handle.
  - Keep USC2 encoding (no change from Windows). UTF-8 would be nice, but not strictly required so drop it.
- Allow server to associate modified behavior on a per-handle basis.


Details

Proposed SMB3 POSIX Extensions

- Negotiate Protocol

Proposed POSIX Extensions

- Create/Open

Proposed POSIX Infolevels

- Query/SetInfo and Query_DIR

```
struct posix_v1_query_inode_info_response {
        /* Returned for context SMB2_POSIX_V1_STAT_INFO */
        __le64 Uid;
        __le64 Gid;
        __le32 Type;
        __le64 DevMajor;
        __le64 DevMinor;
        __le64 UniqueId;
        __le64 Permissions;
        __le64 Nlinks;
} __packed;
```

Statfs (“stat -f”)

```
+struct posix_v1_query_fs_info_response {

        /* Returned for context SMB2_POSIX_V1_STATFS_INFO */

        /* EXISTING posix extensions for fs info is good enough, note For undefined recommended transfer size return -1 in that field */

        __le32 OptimalTransferSize;  /* bsize on some os, iosize on other os */

        __le32 BlockSize; /* f_frsize, disk bytes avail based on this size */

    /* Next three fields are in terms of the block size above. If block size unknown, 4096 would be reasonable block size for a server to report. Note that returning blocks/blocksavail
removes need  to make second call (to QFSInfo level 0x103.  UserBlockAvail is typically less than or equal to BlocksAvail,  if no distinction is made return the same value in each */

        __le64 TotalBlocks;

        __le64 BlocksAvail;       /* bfree */

        __le64 UserBlocksAvail;   /* bavail */

        __le64 TotalFileNodes;

        __le64 FreeFileNodes;

        __le64 FileSysIdentifier;   /* fsid */

        /* NB Namelen comes from FILE_SYSTEM_ATTRIBUTE_INFO call , and flags can come from FILE_SYSTEM_DEVICE_INFO call   */

        /* In Linux  f_type is always 0xFE 'S' 'M' 'B' since that is the fs, not the server's os – so server does not have to return it */
```

POSIX Extensions – Where do we go from here?

- To Redmond next week for continued testing/prototyping
- Timeline, Documentation and Process