File-private Locks

Jeff Layton <jlayton@redhat.com>

POSIX Locks

- Byte-range read/write locks
- Manipulated via fcntl() syscall
- Typically operate in separate namespace from flock() locks (not true on BSD)

The Problems with POSIX Locks

- POSIX locks are “owned” by the process
  - locks acquired by threads within a process cannot conflict with one another!
- released on **any** close of the file
  - leads to the library problem
  - also problematic with symbolic and hardlinks

Conclusion: POSIX locks are useless for any non-trivial program. For NFS, this is especially a problem since POSIX locks are the preferred method for synchronizing file access between applications running on different hosts!

BSD (aka flock()) Locks

- Much more sane semantics…
- “owned” by the open file, not the process
- only released automatically when last reference to open file is released

The only problem…they are whole-file locks.

Why can’t we have both?

- Why not hybridize the two?

- A new type of lock that interoperates with “classic” POSIX locks, but that is “owned” by the open file instead of the process.

File-private Locks (part 1)

Manipulated with fcntl() just like classic POSIX locks, but with new commands:

- `F_GETLKP`
- `F_SETLKP`
- `F_SETLKPW`

Commands are very similar to classic POSIX lock equivalents, and take same `struct flock` as an argument.

File-private Locks (part 2)

- resulting locks will always conflict with classic POSIX locks
- file-private locks are owned by the open file, not the process
- BSD lock-like semantics for inheritance across dup() and fork()
- locks only released when open file is released (on last close, not **any** close)

Patch Status and Plans

- Linux patchset sitting in linux-next now with aim toward merging in v3.14
- Small companion patchset for glibc to add the new cmd values to fcntl.h for Linux
- Currently requires compiling with:

```
#define _GNU_SOURCE
```

- Several open-source projects have expressed interest.
- May become formal part of POSIX?

Further Info

Blog post:

http://jtlayton.wordpress.com/2014/01/07/file-private-posix-locks-aka-un-posix-locks/

LWN article:

http://lwn.net/SubscriberLink/586904/6502e4bfc93c3134/