xFS Kernel Threads Support

Doug Doucette

This document describes the kernel thread functionality and interfaces to be used by xFS.

1.0 Introduction

The xFS project will use kernel threads for its implementation. Threads improve parallelism in the multiprocessor environment. We propose to use thread control interfaces similar to those described in POSIX P1003.4a, Draft 7 (the current draft as of now), for two reasons:

1. We don't have to think as hard about the interface design, compared to designing interfaces from scratch
2. We can more easily run a user-mode simulation of the filesystem, since the thread interfaces match the standard ones for user mode

In effect, this choice makes our upper-level kernel code "look" more like user-level code in structure. The xFS project does not, however, need the full set of interfaces defined in the POSIX specifications.

2.0 Kernel Threads Model

All kernel threads share the same address space: they are part of the kernel "process". We assume that a scheduling model sufficient for the kernel threads underlying user threads is sufficient for xFS kernel threads as well.

We do not believe that it is necessary (for xFS) for the threads package to support essentially instantaneous thread creation. xFS is willing to use a pool of threads and to create or destroy them only when large changes in the system load require a different number of threads be available. In particular, we do not believe that we would implement system calls by creating a thread to handle them.

3.0 Interface Summary

For each section in the POSIX specification we will identify interfaces that we do and do not need. In some cases, not all the functionality will be needed; these cases will be identified (eventually).

3.1 Process Primitives (Section 3)

forkall, **pthread_atfork**, **pthread_kill**, plus changes for **wait**, **_exit**, **alarm**, and signal routines are all not needed, since we are not dealing with processes, just the threads.

3.2 Process Environment (Section 4)

This section covers variables and defined constants that allow the application to determine the system's configuration with respect to the threads implementation. We do not need the interfaces; we may need to **#ifdef** based on the presence or absence of some features.

3.3 Language-Specific Services for the C Programming Language (Section 8)

raise, **setlocale**, **setjmp**, and **longjmp** (and variants) are not needed.

3.4 Synchronization (Section 11)

Semaphore, mutex, and condition variable support is needed. Much of this can be built using current kernel synchronization primitives. The POSIX interfaces should be supplied directly, to allow easier movement of code from user to kernel execution.

3.5 Execution Scheduling (Section 13)

All that xFS needs is scheduling that is priority-based and round-robin (at the same priority level). We do not need preemption for higher-priority threads except at specified (yield) points. This is how the current kernel works, and writing arbitrarily preemptible code won't be worth the effort for xFS.

The interfaces can be used as in POSIX, or other simpler ones are also OK. The POSIX interfaces are pretty complex. One set of features that is complex but we should consider including is the priority inheritance associated with mutexes, to avoid priority inversions.

3.6 Thread Management (Section 16)

This section includes some of the thread attribute routines (**pthread_attr_***xxx*) plus **pthread_create**, **pthread_join**, **pthread_detach**, **pthread_exit**, **pthread_self**, **pthread_equal**, and **pthread_once**. These interfaces are all needed.

3.7 Thread-specific Data (Section 17)

This feature is needed, in some form. Two useful forms of the interface are described in the POSIX specification: a single (void *) pointer per thread, which points to arbitrary per-thread data; or, a key/value mechanism, where an opaque key is associated with each per-thread datum. We can use the single pointer interface instead of the keyed interface, if the threads implementors prefer that. The generality of the POSIX interface is not needed by the xFS project.

3.8 Thread Cancellation (Section 18)

pthread_cancel and its related interfaces are not needed.

3.9 Reentrant Functions (Section 19)

These changes to C library functions are irrelevant to us. The relevant corresponding kernel routines are already reentrant.