RFC: Thread Local Storage Cleanup in Thread-Safe HDF5 for Windows

Dana Robinson

Although not multithreaded internally, the HDF library can optionally be made safe for access by multiple concurrent threads. When enabled, this thread-safe feature uses the designated threading library to implement a global mutex around the library and to provide thread-local storage (TLS) for thread-specific error reporting and function traces. The cleanup of these per-thread resources, however, is incomplete on Windows.

In this RFC, we present a thread clean-up mechanism for Windows and describe its limitations. This functionality will appear in the 1.8.13 release of HDF5.

1 Introduction

The default library build is not safe for access via multiple threads. Library data structures such as the HDF5 file's metadata cache are created on a per-process, not per-thread, basis and access is not protected via a mutex. This can lead to corrupted internal state if a set of operations that modifies a central library data structure is preempted by another thread that also accesses that data structure. Optionally, the library can be built for thread-safe access, which uses a global mutex that ensures all HDF5 API calls are atomic. This is not enabled by default, however, due to the large amount of locking overhead inherent to this scheme.

Although the library does not create threads, the designated threading library[1] is used to implement the global mutex. The threading library is also used to provide thread-local storage (TLS), which has the following purposes in the HDF5 library:

- Maintaining thread-specific error stacks.
- Maintaining thread-specific function trace stacks. This is only enabled when the library is built with this option.
- Maintaining a counter for determining thread cancellability (pthreads only).

In each case, a single TLS slot or key is used for storage.

The original implementation of thread-safe HDF5 was implemented using pthreads, where TLS can be configured to automatically be disposed of as threads exit. When the code was modified to use Win32 threads on Windows, resource leaks were created since TLS cannot be automatically cleaned on that platform.

[1] pthreads on POSIX systems and Win32 threads on Windows.

This mishandling of thread-local storage is the motivation for this RFC.

2 Proposed Solution

2.1 Thread Cleanup in DllMain

On Windows, TLS is usually managed via the library's `DllMain()` function. This function is called whenever processes and threads are either attached or detached. Since the HDF5 library does not have a `DllMain()` function (they are not required), one will be added to H5.c. Internal functions would then added to H5TS.c that create and destroy TLS as necessary.

2.2 No Solution for Static Linkage

Unfortunately, `DllMain()` only exists in dynamic libraries, so this mechanism will not work when the HDF5 library is built as a static library. This means that, for the time being, the static + thread-safe combination will be unsupported on Windows.

3 Recommendation

The `DllMain()` function will be added to both the 1.8 and 1.10 versions of the library. The 1.8 changes would be ready for version 1.8.13, to be released in May 2014.

Acknowledgements

This work is being internally funded by The HDF Group.

Revision History

March 19, 2014:	Version 1 circulated for comment within The HDF Group.
March 24, 2014:	Version 2 incorporates the group's changes. Circulated on the forum.

Glossary, Terminology

This section is not required, but is highly recommended if there are terms in the RFC that may not be familiar to the readers.

thread-safety	A library is thread-safe when it can be accessed from more than one thread of execution. Thread-safety does not imply the creation of threads by the library.
thread-local storage (TLS)	A mechanism by which each thread uses a private copy of a variable. Called thread-local storage (usually abbreviated TLS) in Win32 threads, keys in pthreads.

References

1. Microsoft. "Thread-Local Storage", http://msdn.microsoft.com/en-us/library/windows/desktop/ms686749%28v=vs.85%29.aspx (March 19, 2014).

2. die.net. "pthread_key_create(3) man page", http://linux.die.net/man/3/pthread_key_create (March 19, 2014).