Appendix B. Brief comparison of concurrency libraries

Concurrency and multithreading support in programming languages and libraries aren't something new, even though standardized support in C++ is new. For example, Java has had multithreading support since it was first released, platforms that conform to the POSIX standard provide a C interface for multithreading, and Erlang provides support for message-passing concurrency. There are even C++ class libraries such as Boost that wrap the underlying programming interface for multithreading used on any given platform (whether it's the POSIX C interface or something else) to provide a portable interface across the supported platforms.

For those who are already experienced in writing multithreaded applications and would like to use that experience to write code using the new C++ multithreading facilities, this appendix provides a comparison between the facilities available in Java, POSIX C, C++ with the Boost Thread Library, and C++11, along with cross-references to the relevant chapters of this book.

Feature	Java	POSIX C	Boost threads	C++11	Chapter reference
Starting threads	`java.lang.thread` class	`pthread_t` type and associated API functions: `pthread_create()`, `pthread_detach()`, and `pthread_joint ()`	`boost : : thread` class and member functions	`std : : thread` class and member functions	Chapter
Mutual exclusion	`synchronized` blocks	`pthread_mutex_t` type and associated API functions: `pthread_mutex_lockt)`, `pthread_mutex unlock()`, etc.	`boost : :mutex` class andmember functions, `boost::lock_guard<>` and `boost :: unique_lock<>` templates	`std: :mutex` class and member functions, `std:: lock_guardo` and `std: :unique_lock<>` templates	Chapter
Monitors/ waits for a predicate	`wait()` and `notify ()` methods of the `java.lang.` Object class, used inside `synchronized` blocks	`pthread_cond_t` type and associated API functions: `pthread_cond_wait()`, `pthread_cond_timed_wait ( )`, etc.	`boost:: condition_variable` and `boost:: condition_variable_any` classes and member functions	`std:: condition variable` and `std:: condition_variable` any classes and member functions	Chapter
Atomic operations and concurrency-aware memory model	`volatile` variables, the types in the `java.util.concurrent.atomic` package	N/A	N/A	`std::atomic_xxx` types, `std: :atomic<>` class template, `std::atomic_thread_fence ( )` function	Chapter
Thread-safe containers	The containers in the `java.util.concurrentpackage`	N/A	N/A	N/A	Chapters 6 and 7
Futures	`java.util.concurrent.future` interface and associated classes	N/A	`boost :: unique_future<>` and `boost :: shared_future<>` class templates	`std: : future<>`, `std::shared_future<>` and `std::atomic_future<>` class templates	Chapter
Thread pools	`java.util.concurrent.ThreadPoolExecutor` class	N/A	N/A	N/A	Chapter
Thread interruption	`interrupt ( )` method of `java.lang.Thread`	`pthread_cancel()`	`interrupt ( )` member function of `boost :: thread` class	N/A	Chapter