Subject: Job package: A lightweight worker thread for non-blocking operations.
Posted by Oblivion on Sun, 10 Sep 2017 10:29:46 GMT
View Forum Message <> Reply to Message

Hello,

Below you will find a worker thread implementation which -hopefully- can simplify creating non-blocking or asynchronous applications,
or allow you to easily port your applications to MT-era. I also supplied an example code demonstrating its basic usage pattern.

Job package for Ultimate++

This template class implements a scope bound, single worker thread based on RAII principle. It provides a return semantics for result gathering, functionally similar to promise/future pattern (including void type specialization). Also it provides a convenient error management and exception propagation mechanisms for worker threads, and it is compatible with U++ single-threaded mode.

Note that while Job is a general purpose multithreading tool, for high performance loop parallelization scenarios CoWork would be a more suitable option. This class is mainly designed to allow applications and libraries to gain an easily managable, optional non-blocking behavior where high latency is expected such as network operations and file I/O, and a safe, container-style access to the data processed by the worker threads is preferred.

Features and Highlights

- A safe way to gather results from worker threads.
- Simple and easy-to-use thread halting, and error reporting mechanism.
- Exception propagation.
- External blocking is possible.
- Optional constant reference access to job results.
- Compatible with U++ single-threaded environment.
- All Job instances are scope bound and will forced to finish job when they get out of scope.

Known Issues

- Currently none.

History

- 2017-10-07: Compatibility with U++ single-threaded mode is added.

- 2017-10-01: Global variables moved into JobGlobal namespace in order to avoid multiple definitions error. Accordingly, global functions are defined in Job.cpp.

- 2017-09-22: Exception propagation mechanism for job is properly added. From now on worker threads will pass exceptions to their caller.
Void template specialization is re-implemented (without using future/promise).
Constant reference access operator is added. This is especially useful where the data is a container with iterators (such as Vector or Array).

- 2017-09-19: std::exception class exceptions are handled, and treaated as errors. (For the time being.)
void instantiation is now possible.
Jobs will notify their workers on shutdown.
Clean up & cosmetics...

- 2017-09-18: Clear() method is added. Worker id generator is using int64. Documentation updated.

- 2017-09-17: Future/promise mechanism, and std template library code completely removed. From now on Job has its own result gathering mechanism with zero copy/move overhead.

- 2017-09-16: Job is redesigned. It is now a proper worker thread.

- 2017-09-10: Initial public beta version is released.

Hope you'll find it useful.

Best Regards,
Oblivion

File Attachments

1) Job Package and Examples.zip, downloaded 489 times