Subject: Re: RE: Job package: A scope-bound worker thread for non-blocking operations.
Posted by Oblivion on Sun, 08 Oct 2017 13:01:15 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I wrote a prototype for U++ implementation of promise/future mechanism. It is 104 LOCs, and it took an hour for me to to design and write it (It is in its current state not perfect but it works.)

It is consisted of three classes (please do not dwell on naming, I didn't think them through, they'll change):

- WorkEntry -> std:promise   (This class is not explicitly called. It is allocated on the heap (using new), and its ownership is passed to WorkResult.)
- WorkResult -> std: future (With pick semantics, void type specialization, error handling, exception propagation, and optional constant reference access to the result.)
- Work   (Contains a CoWork instance, and exposes Do() and Finish() methods.)

I re-implemented your Async() code in Work class, using CoWork::Do() instead of CoWork::Schedule(). The rationale behind this decision is that sometimes asynchronous operations need to be externally blocked, all at once ("wait for all works to finish"). And I found no other reliable way. Also, works use a single semaphore to wait and run.

Example code:

```
// Error handling, picking/moving.
 auto r1 = work.Do([=] { throw Work::Error("Worker failed."); });
 work.Finish();
 auto r2 = pick(r1); // r1 is picked.
 if(r2.IsError()) {
  Cout() << r2.GetErrorDesc() << '\n';
 }
```

There is room for improvements, and probably some issues I didn't encounter yet or I overlooked. Feel free to comment on it. (One area that needs improving and to be made more error resistant is moving/picking behaviour. I'll improve them)

Below you can find the Work, and WorkExample, which is basically JobBencmark adapted to Work.

Best regards,

File Attachments

1) Work.zip, downloaded 447 times