
Subject: AsyncWork

Posted by [mirek](#) on Sat, 14 Oct 2017 09:34:45 GMT

[View Forum Message](#) <> [Reply to Message](#)

AsyncWork is U++ take on std::future mechanism. The difference is that AsyncWork is using CoWork thread pool as backend and, more importantly, allows for cancelation of job.

```
#include <Core/Core.h>
```

```
using namespace Upp;
```

```
CONSOLE_APP_MAIN
```

```
{  
  StdLogSetup(LOG_FILE|LOG_COUT);
```

```
  auto a = Async([](int n) -> double {  
    double f = 1;  
    for(int i = 2; i <= n; i++)  
      f *= i;  
    return f;  
  }, 100); // Schedules job to be executed by threadpool, returns AsyncWork for the return value  
  and job control
```

```
  DUMP(a.Get()); // Makes sure job is finished (can execute it if it has not started yet), returns the  
  result
```

```
  auto b = Async([] { throw "error"; });
```

```
  try {  
    b.Get(); // exception is propagated  
  }  
  catch(...) {  
    LOG("Exception has been caught");  
  }
```

```
  auto c = Async([] {  
    for(;;)  
      if(CoWork::IsCanceled()) {  
        LOG("Work was canceled");  
        break;  
      }  
  });  
  Sleep(100); // make it chance to start  
  // c destructor cancels the work (can be explicitly canceled by Cancel method too)  
}
```
