Project #2b – Asynchronous Tasks

Purpose:

Asynchronous tasks are requests for processing that run on a thread different than the caller's thread. This allows the caller to continue on with its activities while the requested processing runs concurrently. Eventually the caller may need the results of that background processing to continue and may need to wait for the results. An asynchronous request may be made as a function call or by sending a message. For this project you will experiment with both asynchrounous function calls and message passing.

Asynchronous requests imply communication that may be: one-way, return a future value, or trigger a callback to return results. This project invites you to use the Tiny Web Client and Server of the previous project to implement spawning of tasks across process, machine, and/or technology boundaries, supporting each of these three communication modes. Alternately you may implement Asynchronous Tasks with a Task class using a thread pool, modeled after the .Net Task Class. You will be required to do some performance comparisons between your implementations and platform specific methods of asynchronous programming[1]. Either of these alternatives are to be implemented on both Windows and Linux.

As part of this project you will prepare and deliver three presentations. The first describes the technologies you are using and gives a brief description of the application you will implement. The second presents several probing projects that illustrate how the technology works. Finally, the third presentation shows your design, implementation, and demonstrates your project's capabilities.

Requirements:

For your Asynchronous Tasks project you:

1. **shall** use standard C++ and the standard library, compile and link from the command line, using g++ within the NetBeans or Eclipse IDE and Visual C++ in the Visual Studio IDE.

2. **shall** implement either a. or b.:
   a. develop HttpClient and HttpServer, based on sockets, using the HTTP 1.0 protocol[2], as described in Project #2a. You are not required to support both ip4 and ip6 with the underlying sockets package as is required for that project.
   b. Implement a Task class that uses a Thread Pool to run Tasks asynchronously.

3. **shall** create a test client that demonstrates spawning of Tasks and measures their performance under various jobs and number of concurrent tasks.
   a. For message-passing it specifies, via message header attributes the requested activity and one of the communication modes: one-way, return results in a reply message, or return a reply message that invokes a callback function, specified in a reply message header attribute. This will require including a blocking queue for received messages and developing a dispatch mechanism on the client for replies that invoke callbacks.
   b. For function invocation it specifies, via function arguments the requested activity and on of the communication modes.

4. Each of these invocations **shall** be timed with the a high resolution timer[3].

5. **shall** explore and develop timed demonstration code for several of the native asynchronous technologies on Windows[4] and Linux[5]. Of particular interest is I/O completion ports on Windows[6]. Compare the performance results of your implementations with the native technologies.

---

[1] If you include interprocess requests you are not required to implement a thread pool. If you don't then you are required to implement a thread pool or use a platform provided thread pool.
[2] http://en.wikipedia.org/wiki/HTTP, http://en.wikipedia.org/wiki/List_of_HTTP_status_codes
[3] http://www.lcs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/HiResTimerNativeCpp/
[4] http://msdn.microsoft.com/en-us/magazine/jj721588.aspx,
http://kennykerr.ca/articles/,
http://msdn.microsoft.com/en-us/magazine/jj883951.aspx
[5] http://www.ibm.com/developerworks/linux/library/l-async/,
[6] http://msdn.microsoft.com/en-us/magazine/hh580731.aspx,
http://msdn.microsoft.com/en-us/magazine/jj883951.aspx

Essentially, your goal is to explore implementation of asynchronous processing using platform resources, C++11 facilities, and message passing, possibly based on an HttpClient and HttpServer implementation, comparing performance and ease of use.

You will find it helpful to look at the Man pages for System Calls on your Linux system. Those describe the semantics of each call and the header files you will need to include.