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 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 requires you to use the Tiny Web Server of the previous project to implement spawning of tasks across process, machine, and/or technology boundaries, supporting each of these three communication modes. You will be required to do some performance comparisons between your implementations and platform specific methods of asynchronous programming.

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** develop HttpClient and HttpServer, based on sockets, using the HTTP 1.0 protocol[1], 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.

3. **shall** create a test client that uses HttpClient. The client constructs HTTP messages that request one of several processing activities to be carried out on the server. 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.

4. **shall** create a test server that uses HttpServer. The server provides a blocking queue that collects incoming messages. It parses client request messages and dispatches the requested processing. When the processing completes it sends a reply message that matches the mode of the incoming request.

5. **shall** define processing pShort, that requires very little time to complete, and define processing pLong, that requires several seconds to complete, then invoke pShort and pLong using each of the three communication modes in an HTTP channel constructed from HttpClient and HttpServer. Each of these invocations **shall** be timed with the a high resolution timer[2].

6. **shall** explore and develop timed demonstration code for several of the native asynchronous technologies on Windows[3] and Linux[4]. Of particular interest is I/O completion ports on Windows[5].

---

[1] http://en.wikipedia.org/wiki/HTTP, http://en.wikipedia.org/wiki/List_of_HTTP_status_codes
[2] http://www.lcs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/HiResTimerNativeCpp/
[3] http://msdn.microsoft.com/en-us/magazine/jj721588.aspx, http://kennykerr.ca/articles/, http://msdn.microsoft.com/en-us/magazine/jj883951.aspx
[4] http://www.ibm.com/developerworks/linux/library/l-async/,
[5] http://msdn.microsoft.com/en-us/magazine/hh580731.aspx,

7. **shall** explore and develop timed demonstration code using the new thread facilities in C++11, especially async and packaged_task facilities[6].

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

---

http://msdn.microsoft.com/en-us/magazine/jj883951.aspx

[6] http://en.cppreference.com/w/cpp/thread