Project #2e – Cross Platform Infrastructure

due last day of class

Purpose:

This project develops a set of cross-platform support packages for Windows and Linux. The set provides services to application programs that encapsulate operations needed from the operating system. The services have the same application interface on Windows and Linux with internals that use the platform APIs. The intent is that all application code above the service packages can be common to both Windows and Linux[1].

The core services include: file system[2] management, sockets and communication, process management, threads[3], locks, and blocking queues. This project will also provide XML processing for both platforms since that is not provided by native C++ libraries.

Requirements:

Your CoreServices project:

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

2. **shall** provide service packages for file system management[4], sockets and Communication[5], process management, specialized locks[6], blocking queues[7], and processing XML[8].

3. **shall** provide an application that runs on both Windows and Linux that requires all the services developed for this project. You may wish to consider using the core processing provided by Project #1.

4. **shall** provide a native unit test framework[9] that you use to thoroughly test each of the packages developed as part of this project.

A significant part of the credit you earn for this project is based on the effectiveness and esthetics of your summary application.

You will find the following reference very useful in understanding how the HTTP protocol works:
http://www.jmarshall.com/easy/http/

---

[1] Look at the Cross-Platform GUI project to see how this design concept can be extended to GUI-based applications.
[2] These already exist in code I developed here:
http://www.lcs3.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/.
[3] Threads and locks can be provided directly from the C++11 library – no need for additional development.
[4] This is already almost complete: http://www.lcs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/
[5] You may use my sockets package as a starting point:
http://www.lcs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/
[6] C++11 provides a set of locks but does not implement a slim reader/writer lock, so we will.
[7] You may start with my condition-variable based blocking queue for Windows: ??
[8] I've implemented XmlReader and XmlWriter packages for windows here:
http://www.lcs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/
[9] See the FileSystem project here: http://www.lcs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/