Project #4 – Remote Code Management Facility

due Tuesday, April 28

version 2.2

Purpose:

This project requires you to design and implement a facility for remote code management (RCM). Management consists of transferring files to and from the local and remote machines, exploring a directory subtree on a remote machine, identifying files and searching for specific texts within files. The facility will use communication software you built in Project #3 to start search tasks on a remote server and return results while serving multiple concurrent clients.

Requirements:

Your Message-Passing Communication project:

1. **shall** use standard C++ and the standard library, compile and link from the command line, using Visual Studio 2013, as provided in the ECS clusters and operate in the environment provided there[1].

2. **shall** use services of the Windows Presentation Foundation (WPF), using either C# or C++/CLI, for all input and output to and from the user's console and C++ smart pointers for all server dynamic memory management.

3. **(3) shall** provide a server application that supports uploading and downloading source code files[2], exploring one specific directory subtree, and performing file searches and concurrent string searches[3] within source code files in that tree.

4. **(3) Shall** provide a client process that uses Windows Presentation Foundation (WPF) to build a Graphical User Interface (GUI) that supports file transfer and processing requests and displaying results and performance information.

5. **(3) Shall**, for text search commands, accept one or more end-point addresses to specify machines to search, wild cards to specify file patterns to match, and regular expressions to specify text to find.

6. **(4) Shall,** in response to a text search command, return a list of files that contain the text and the paths and machines on which they reside.

7. **(4) Shall**, in response to a file search command, passing a path[4] and file patterns, return a list of files with their paths found in the directory subtree rooted at the specified path. The client **shall** provide the option[5] to return an XML string showing the entire directory subtree containing matched files.

8. **(1) shall** support concurrent processing of text searches, including the capability to specify, at the client, the number of threads that will participate in that server processing.

9. **(1) Shall** enable the measurement of time required to carry out a processing request and also the end-to-end time to request a processing task and receive the corresponding reply. Please display the results in milliseconds.

10. **(1) shall** use the high resolution timer provided by the std::chrono library.

11. The GUI client and server **shall** display information to demonstrate clearly and succinctly that all requirements of this project have been met[6].

12. **shall** provide one compile.bat and one run.bat file that build and then execute your demonstrations Please package your project code, projects, solution, and batch files in a single zip archive. Please do not submit archives that are not zips.

---

[1] VC++ 2013 is available in all the ECS clusters.

[2] Note that no check-in, check-out, versioning, or dependency processing is required for this project, although those capabilities make sense for code servers.

[3] A 5 point bonus will be awarded for correctly implementing both search and scope analyses with concurrent processing.

[4] This path is relative to the server's storage root path.

[5] This option can be implemented with a radio button or checkbox. The only thing the option does is to show all the directories in the subtree but only the matching files.

[6] A good way to do that is to display a background console with response outputs demonstrating the requirements as you meet them. You will be shown how to do that in class. No points are associated with this requirement since it is telling you that you won't get points for requirements you don't clearly show you have met.