

**Fall 2006: CS 431/631/731 Distributed Computing
Homework-2**

200 points. Individual Work Only. Due October 9, 2006 before class.

Objective:

Design and implement a multi-threaded client-server program in Java using Java RMI for communication between the client and server programs.

Description:

Instead of using sockets for implementing the File Server program in Homework-1 use Java RMI for communication. The various commands that the client and server can process along with the action performed at the server is given below:

LIST – Send list files in the current directory on the server (initially the directory where the server program was started would be the current directory)

SIZE *filename* – Send file size of the specified file name, a negative number if file does not exist

GET *filename* – Send the specified file to the client process; if the specified file does not exist on the server then the server send an error message (a negative #); filename is assumed to reside in the current directory on server and copied to the current directory on the localhost.

PUT *filename* – Receive the specified file from the client; filename is assumed to be residing in the current directory on client and is copied to the current directory on the server.

CD *newdirectory* – change to specified directory on the server; print an error message if the directory does not exist.

QUIT – client program exists after closing any open connections, no action is performed on the server.

Assume that the files are in binary format for file transfer. Use the following sample input files to test your program:

<http://www.cis.uab.edu/cs632/software/homework0.c>

<http://www.cis.uab.edu/cs632/software/homework2.tar>

<http://www.cis.uab.edu/cs632/software/image1.ppm>

<http://www.cis.uab.edu/cs632/software/image2.ppm>

Each user will be assigned specific port # to start the server process. The server process must be started before starting the client process. For initial testing and debugging purposes you can start the server and client process on the same machine after that you can start the server on a different machine.

Graduate Students Only (Bonus for Undergraduates):

In addition to the server being multi-threaded, extend the client program to be multi-threaded as well. Specifically, commands GET and PUT must accept multiple filenames (maximum of 5) as arguments and use a separate thread to perform the GET and PUT operation for each file.

Working Environment:

You can develop and test the client and server programs on any machine that you have access to. If you wish to use CIS machines then you must use vulcan[1-8].cis.uab.edu (Linux workstations in the undergraduate lab). There is no need to be physically in the lab to use these machines, you can connect to these machines remotely using an SSH Client. For more details on how to connect to these machines using SSH check out sample screen shots and a Flash demo provided at: http://www.cis.uab.edu/cs333/CIS_UNIX_howto.html.

**Fall 2006: CS 431/631/731 Distributed Computing
Homework-2**

You need to add the following lines to the end of .bash_profile file to have the correct Java environment:

```
export JAVA_HOME=/netbin/java
export PATH=${JAVA_HOME}/bin:${PATH}
```

After adding the above two lines, logout and login again. If you type “which java” you should see: /netbin/java/bin/java.

Short Answer Questions (no need to implement any of this):

Assume that the server program is running on multiple machines. What changes, if any, are required to the server and client program if we were to access files on any of these machines using the following naming convention: hostname:directory/filename. For example:

```
GET vulcan1.cis.uab.edu:/home/puri/cs631/file1
PUT vulcan2.cis.uab.edu:/home/puri/file2
GET vulcan1.cis.uab.edu:/home/puri/cs631/file1 vulcan2.cis.uab.edu:/home/puri/cs631/file2
```

Write down the changes required, if any, for both server and client program.

Feedback Questions (answer to these questions has no impact on your grade):

Was this homework too difficult, or too easy?
Was the assignment fun or challenging?
Was there something that was unclear?
Was the homework too long for the given amount of time?
What did you learn from this homework?

Submission Instructions:

List ALL the references you used in this homework as well as test cases used to test your programs. This includes any classes that you used that you did not write and any help you received from any other sources. Use appropriate class name and include comments to indicate various operations performed by the program. Your program must have the following header information within comments:

```
/*
  Name:
  BlazerId:
  Homework #:
*/
```

Make a directory CS631/homework2 and then two separate directories one for the server (server) and one for the client program (client). Following files must be included in the Server directory:

1. Server Service Interface Definitions
2. Server Service Implementations
3. Skeletons for the Server Implementations
4. Stubs for the Server Implementations
5. Stubs for the Client Implementations

**Fall 2006: CS 431/631/731 Distributed Computing
Homework-2**

Following files must be included in the Client directory:

1. Client Service Interface Definitions
2. Client Service Implementations
3. Skeletons for the Client Implementations
4. Stubs for the Client Implementations
5. Stubs for the Server Implementations

Create a tar/zip file of the homework2 directory using the following filename format: blazerId-cs431-hw2.tar or blazerId-cs431-hw2.zip (graduate students use 631 or 731 instead of 431). The tar/zip file must include the source code, txt/word/pdf/ps document for the typed solution to short answer questions and the feedback questions, instructions for executing the programs. Login to WebCT and go to the assignments section (Homework-2) and upload the tar/zip file (using the browse button under attachments). There is no need to turn in any printed solutions for this homework, WebCT submission is sufficient.

Late Submissions:

Submissions must be made on the due date before the beginning of the class. Late submissions will lose 10% for every 24-hour period, up to a maximum of 50% (weekends and holidays count as one 24-hour period). Any submissions made after one-week will receive a score of 0 for this homework.

Resources:

1. Java Developer's Almanac - Code Samples:
<http://java.sun.com/developer/codesamples/EXAMPLES/>
2. Java RMI Tutorial: <http://java.sun.com/developer/onlineTraining/rmi/RMI.html>