CIS 4930 Internet Networking Technologies, Spring 2023
Programming Assignment #1

Due: Please submit your source codes and report (**in PDF format)** via CANVAS by 10:40am (**firm deadline**) on Monday February 27th

Introduction

In this programming assignment, you will develop a simple client/server implementation for socket communication. It is intended for the students to experience socket programming with proper exception handling. The implementation will have a server program and a client program running on two CISE machines. They communicate with each other using socket communication on the TCP/IP protocol suite.

Your server process must store three (3) jokes at the server machine. Jokes should be in plain text format, i.e., with the txt file extension. After initialization, the server process listens and responds to a client's request. A client will try to connect to the server and send out a command like "Joke 1". The command means to retrieve a joke, and the number is "1". Upon receiving the command, the server will respond to the client with the corresponding file, i.e., joke1.txt. More details will be explained later in the following sections.

Prerequisites

Socket

A socket is the endpoint of a two-way communication link between two programs running on the network. It is bound to a port number so that the TCP layer can identify the application host that data are destined to/from. The socket API allows application programs to communicate remotely over the Internet. Socket send/receive primitives correspond to the write/read operations in a file system. In this project, we will use sockets over TCP/IP in order ensure the reliability of the communication.

More about socket communication can be found from the following links (checking out the books in library and doing some internet search are also encouraged):

[Java] http://www.oracle.com/technetwork/java/socket-140484.html
http://en.wikipedia.org/wiki/Berkeley_sockets

Specification

Server Program

A server program is an always-running process that accepts the clients' requests. This program starts earlier than all clients and waits for the in-coming connection. After accepting an incoming request from a client, it responds with a message "Hello!" to the client. Then whenever it receives a command from the client, it prints out the command on screen and then returns with the file/object (e.g. for the message "Joke 1" from client, the server will send a file "joke1.txt" to the client). If the server receives a command "bye" from a client, it closes the corresponding socket.

Client Program

A client will be started on a different machine after the server is running. After connecting to the server and receiving "Hello!" message, client program prints it out and lets the user input a command line. Then it will send the command to the server and receive a response. After writing out the joke as a file, it will let the user input again. The following are the operation commands used in this assignment.

Joke 1

Joke 2

Joke 3

Exception Handling

Your server program should be able to handle unexpected inputs. For example, consider a command "Joke $%". This operation cannot be done. In this assignment, the server needs to send an error message instead of the wrong result. The client program shall print out the corresponding error message after receiving it.

Termination

Your programs shall terminate gracefully. If there are runaway processes, points will be deducted. The detailed process is: when user types the command "**bye**" and the client send it to the server, server will reply with a "disconnected" message and exit. Upon receiving "disconnected" message, the client process will print out "exit" on screen and exit.

Note on Run-away Processes for the Graceful Termination:

While testing your programs, your run-away processes could exist. However, these run-away process must be terminated. Please check frequently if there is any remaining process after termination. CISE Department has a policy regarding this issue and your access to the department machines might be restricted if you do not clean these processes properly.

Some useful Linux/Unix commands:

To check your running processes: ps -u <your-username>
To kill a process: kill -9 pid
To kill all Java processes: killall java
To check processes on remote hosts: ssh <host-name> ps -u <your-username>
To clean Java: ssh <host-name> killall java

Execution format

Server:
[Java]
java server [port_number]

Client:
[Java]
java client [serverURL] [port_number]

Examples of compilation and execution (do not use the same port number)

[Java]
storm> javac *.java
thunder> javac *.java
storm> java server 21234
thunder> java client storm.cise.ufl.edu 21234

S> ./server 21234
S> get connection from … (IP)
S> get: "Joke 1", return: "joke1.txt" file

C> ./client storm.cise.ufl.edu 21234
C> receive: Hello! (user input: "Joke 1")
C> receive: joke1.txt

Port number

You have to be careful when using a port number, i.e., some port numbers are already reserved for special purposes, e.g., 20:FTP, 23:Telnet, 80:HTTP, etc. Our suggestion is to use the last 4-digit of your UFID as the port number to avoid potential conflicts with other students. Keep in mind that the port number must be less than 2^{16} (=65,536).

Getting the most points

1. The source codes need to be tested on CISE Linux/Unix machines, because we are going to grade your source codes on Linux systems. Please state your testing machines and results in your report.

2. Please try to complete the socket communication part first. After testing that, your program will be able to correctly send a message to the server and receive a response.

3. Try to finish the simplest operations first (e.g. "Joke 1").

4. We will use the "more" command on Linux systems to view your jokes.

5. Have backups whenever your program can do more. You can avoid the situation where your program used to work but not anymore (e.g. after trying to implement exception handling).

Reminder

1. For your programming conveniences, your server program **does not** need to be implemented with thread functionalities (multi-threading) in this assignment. In other words, your server program doesn't need to handle concurrent requests by several clients, i.e., we will test your server program by one client ONLY.

2. All assignments must be done individually. Using blocks of code from other people or any other resources is strictly prohibited and is regarded as a violating of UF Honor Code! Please refer to the UF honor code if you are unfamiliar with it (http://itl.chem.ufl.edu/honor.html).

Report

Submitted report file should be named "**report.pdf"** and include the following:

- Your personal information: Full name, UF ID, and Email
- How to compile and run your code under Linux environment.
- Description of your code structure.
- Show some execution results.
- Lesson learned
- Any additional comments.

Submission Guidelines

1. The name of a source code for the server program should be named “server.java” and the client program “client.java”.
2. Only submit your Java source code files and report, do not include .obj/.class or executable files in your submission.
3. Include “makefile” if you have one. (Not required)
4. Include the report in PDF format.
5. Zip all your files into a packet: Firstname_Lastname_ID.zip
6. We recommend you upload the zip packet as attachment in CANVAS 30 minutes before deadline.

Grading Criteria:

Correct Implementation / Outputs	50%
Readability / Comments / Code structure	20%
Graceful Termination / Exception handling	20%
Report	10%
Total:	100%