School of Computing

Bachelor of Science in Computing.

Programme Code: DT211/3

2012 – 2016

Network Programming Document

Lecturer: Mark Deegan

Student's Name	James Wilson
Class Group	DT211/3
Assignment Number	2
Assignment Title	Simple Server
Date Issued	17/1/2015
Date Due For Return	6/3/2015
Date Returned	6/3/2015

Summary of Assignment

In this assignment I had to build a server and a client using two programming languages Java and C. The client was able to write a message to the socket, the server reads the reply from the socket and it display the reply on the console using a TCP connection. An important aspect to note from this program, was the it had to be robustness (meaning error checking) for example if there is an invalid port number entered or two little or two many arguments passed it would produce an error. Also, the server and client of different programming languages should also be able to write messages through a socket connection to each other.

How to run the programs

Why I used Ubuntu & my directory Structure

In this assignment I installed Ubuntu on my laptop using it as a guest OS in VirtualBox as the program which I was testing in Windows 7 wouldn't compile due to the libraries that need to be imported. This was a work around for the winsock .lib I would of have to of used if I was compiling the program in Windows.

Bin Directory

My directory structure that I used had three folders the binary folder, the source folder and the documentation folder. What I stored in the bin folder was my .o and .class files that were created once the .c or .java was compiled (it will be empty as it will be tested by other users when they run it my make file).

Src Directory

In my source directory, I had multiple sub-directories, which included my java folder, c folder and finally my bash folder. In my java and C directory I stored all the source files being the SimpleJavaServer.java file etc, and the same for the C file along with the Readme file. My bash folder is the fulcrum where the files are compiled by the user via console and run from this directory via user interaction with the console.

Doc Directory

In my documentation directory I had the design document, main document, and the testing document. The design document had a overview of how I was going to develop the program during the course of the three weeks, with some alterations along the way. in my testing document, it consisted of how I was to test my programs, both C and java.

How I compiled My programs & run them

In this assignment, I was having trouble running the c files (and at this stage I did not run test the java files) and installed Ubuntu as mentioned. I compiled in the command line and used getit (Ubuntu's text editor) as I found it useful to read and write my code, and then ran it from command line. As I got the C functional I made a make file to compile the C programs both client and server as I learned via the lecture, then focused on compiling the code issued by Mark Deegan. I again made the make file after I was finished debugging and editing the Java code.

Make File (Compiling files)

I have a make file, which compiles both the java & c files. I also wrote bash scripts that execute the compiled files and store the .class file for the java and the .o file for the c in their appropriate directories in the binary folder. For example, instead of testing /running the program to keep recompiling the changes using for example

```
#C EXAMPLE
SimpleServer.o: ../c/SimpleServer.c
        gcc -o ../../bin/SimpleServer.o ../c/SimpleServer.c
#JAVA EXAMPLE
SimpleJavaServer.class: ../java/SimpleJavaServer.java
        javac -d ../../bin/ ../java/SimpleJavaServer.java
```

that were made by the user (like C Client & C server) you can just run the make by entered the make command into the console.

Bash File (Running files)

Also, in the bash directory I have made bash files which run the actual program that I was developing, I have runSimpleJavaServer which runs the JavaServer program and it passes through the parameters that are stored in the file for example:

```
java -cp ../../bin ie.dit.student.wilson.james.SimpleJavaServer  4444
```

Here I am running the java file with the correct package structure so that the class file will be stored in the binary directory, there in the package directory which is used for Java programming.
In my C bash file for example my Simple C Server I also have a run script which basically again runs the Server and it passed through the respected parameters.

```
../../bin/SimpleServer.o 4444
```

Similar to that in java but C does not include a package structure.

What I have learned & gained from this assignment

From this assignment I firstly have learned how to use Ubuntu operating system and grew very comfortable with the command line. I pushed and pulled from the command line as I found it a lot more efficient rather than using the github app. I further enhanced my knowledge on socket connections from the first assignment where I dealt with a function of socket connections. I also learned the input stream and output stream between the client and the server, and how to read from a buffer (read by line) and display the message on the console. I was comfortable exchanging between two different programming language. I also used printwriter instead of a buffered writer because typically you'd use it for printing to the console.

I also found that I was pushing more frequently to github than the last assignment, as my VM has cashed and github has been helpful in this case and I hope to use github more often than not to develop other assignments.