COMP202-17B - Intro to Java Sockets and Linux

July 18, 2017

1 Introduction

This practical will introduce to you programming with Java in the Linux environment. You will learn about writing, compiling, and executing Java sockets programs in Linux.

When writing code during this course you can use whatever text editor you prefer. The programs you will write do not require you to use a fancy Java IDE; you should be able to use a text editor such as `emacs` or `gedit` and a Terminal to do your work.

Sit down at a computer and login with your own username and password. When you login, navigate to the Terminal application by selecting Applications, Accessories, Terminal. By default, the Terminal starts in your *home* directory, the place where your own files can be created and stored. Type `pwd` to print your current working directory. The `ls` command lists the files in the current directory – try it. Make a new directory called 202 by issuing the command `mkdir 202`. Protect the contents of your work by executing the command `chmod 700 202`. This command ensures only you can read and write files to this directory. Change into the directory by typing the command `cd 202`. Check that you are in this directory by issuing the `pwd` command. Create a new directory in your 202 directory by issuing the command `mkdir prac1` and change into this directory using the commands you have learned so far.

As this practical involves the InetAddress and Socket classes, please time some time to read the documentation on them. You can find the documentation at `http://java.sun.com/javase/6/docs/api/`

2 Academic Integrity

The files you submit **must** be your own work. You may discuss the assignment with others but the actual code you submit must be your own. You must fully also understand your code and be capable of reproducing and modifying it. If there is anything in your code that you don't understand, seek help until you do.

You **must** submit your files to Moodle in order to receive any marks recorded on your verification page.

This assignment is due July 31st 2017 by 11am and worth 6% of your final grade.

3 Resolving IPv4 addresses

In the first part of your practical, you will write a small program to obtain the IPv4 address of a corresponding DNS name. Create a new source code file in the Terminal by typing `gedit resolve.java` and pressing enter. Gedit will open and you can start working on your program.

Your first task is to create a program which resolves a list of names provided on the command line to their corresponding IPv4 addresses. For example:

```
$ java resolve
Usage: resolve <name1> <name2> ... <nameN>
$ java resolve www.twitter.com www.facebook.com invalidname.waikato.ac.nz
www.twitter.com : 199.59.150.7
www.facebook.com : 179.60.193.3
invalidname.waikato.ac.nz : unknown host
```

To create your program, you will need to import the java.net library, and use the InetAddress class. Make sure you catch all exceptions and ensure that your program prints a usage statement if it is not provided with any command line arguments. The output should look exactly as above (unless the IP addresses for any of these names changes before you complete this part!)

When you are ready to test your code, save and exit the file, and then type `javac resolve.java`. If your code is syntactically correct, it will compile without emitting any errors or warning. If any errors are printed, please correct them. When your code compiles without emitting warnings or errors, please type `java resolve` to see that it runs, and then try resolving a few addresses.

Run your program a few times. One of the names in the above example will show a different address each time you run it. Which name is it? How many different IP addresses does it have? Why do you think this name has multiple addresses?

Use the `ls` command to inspect the available files in your directory. What file did the `javac` command produce?

4 Reverse resolving Internet addresses

In the second part of your practical, you will write a small program to obtain the DNS names corresponding to a list of IP addresses. Make a copy of your resolve.java file with the command `cp resolve.java reverse.java` and then `gedit reverse.java` to change the program so it now works as follows:

```
$ java reverse 130.217.250.39 130.217.250.13 127.0.0.2
130.217.250.39 : sorcerer.cms.waikato.ac.nz
130.217.250.13 : voodoo.cms.waikato.ac.nz
127.0.0.2 : no name
```

Make sure you change the class name of your program so it matches the name of your source code file (reverse) or you will not be able to run your code. Also make sure that when you specify an IP address without a name, your code says so, rather than print, for example:

```
$ java reverse 127.0.0.2
127.0.0.2 : 127.0.0.2
```

To accomplish this, you will need to use the compareTo method available with the java String class.

5 Resolving IPv6 addresses

In the third part of your practical, you will extend your resolve.java program to obtain all IP addresses, including IPv6 addresses, associated with a DNS name. Copy resolve.java to resolveall.java. Then, use the method described in the InetAddress class documentation to obtain all IP addresses associated with the name. Iterate through the array which was returned. Run the program on the same set of domain names as before.

Which names have IPv6 addresses?

Which names have more than one IPv4 address? Did you see any more addresses for that name than you saw in the first part of the practical?

6 Simple network server

Your final task is to create a simple network server which will greet each client that connects and then disconnects them. Create your network server by executing the command `gedit SimpleServer.java`. Create a ServerSocket, allowing it to bind to the first free port. When each new connection is accepted, obtain the IP address of the source; resolve the address to its name and then great the client.

Create a network client to go with your server. All your client has to do is connect to the server, print out whatever the server sends back, and then exit. The exchange should look like this:

```
$ java SimpleClient cms-r1-18.cms.waikato.ac.nz 1234
Hello, cms-r1-17.cms.waikato.ac.nz.
Your IP address is 130.217.252.17
```

You should be able to login to a neighbour machine and check that you can communicate with that machine over the network.

VERIFICATION PAGE

Name: _____

Id: _____

Date: _____

Note: this practical should be verified by Monday 31st of July 2017 at 11am. It will be marked out of 6 and is worth 6% of your final grade.

You must submit your code to the Moodle assignment page after you have had the assignment verified. Do not zip your source code. Submit all the individual *.java files.

1. If you were to create a Java program named `foo.java`, what file would `javac` produce?

2. Show the Java documentation page for the java.net.Socket class.

3. How do you tell a ServerSocket to bind to the first free port?

4. Which DNS name in your first task returns a different IPv4 address with each invocation. How many IP addresses does it have? Why?

5. Which DNS name in your second task returns an IPv6 address? What is it? What is the size of the address, in bits?

6. Explain to the demo the structure of your SimpleClient and SimpleServer programs.

7. Allow the demo to read and test your resolveall code.

8. Demonstrate that your simple network server can communicate with another machine in the lab.

9. Show the demo that your code is well commented.