COMP260 – Distributed Programming Worksheet 10

Hosting applications remotely

Introduction

The goals of this worksheet are two-fold, firstly we are going to set-up accounts on a remote Linux server that you can then use to host the Chat Service from week 4. Secondly, you can work towards hosting your MUD servers remotely as part of assignment 2.

How to Linux servers

You all have user-level accounts on a remote server (46.101.56.200). I have everyone's account details and will give them out as required. To connect to this server, I recommend PuTTY (https://www.putty.org/). This can be installed on an external drive and used on your lab PCs. PuTTY will allow you interact with the server machine using a command-line interface. If you are unfamiliar with Linux, we have provided this link in the assignment http://mally.stanford.edu/~sr/computing/basic-unix.html to get you up to speed. Also, learning material is available on Pluralsight and Lynda. However, you do not require a deep understanding of Linux command line for the assignment.

To upload development files to the server, install filezilla (https://filezilla-project.org/) and FTP them from your PC.

I have installed Python 3 (3.5.2) and your applications should run off that. To install your python apps onto the remote server:

1. Load filezilla and log onto the server with your account details using secure ftp. The host is: sftp://46.101.56.200.
2. Copy your .py files onto your file space on the server. Take care to avoid transferring anything that is not a .py file and be aware that Linux is case sensitive
3. Start up a terminal session on the server using PuTTY
4. Run your python application from the command line using python3 <filename>.py

Be aware that your network applications will need to run using the ip address and port for the server, not your local loopback (127.0.0.1). Also, your client will need to use this address and port combination.

You will need to allocate ports between all of you. Currently, ports 9000-9500 are open and these should be enough. Make sure that you do not use ports that you have reserved for other students.

By default, your application will stop executing when you close the PuTTY session that you are running your app from. To get around this, you can use the nohup command to keep the application run on shell exit. The format for nohup is:

```
nohup python3 <filename>.py &
```

To stop a app that is running through nohup, use ps to determine the ID of the app and then kill the task with kill <process_id>, look at this website for more detail: https://www.booleanworld.com/kill-process-linux/

Hosting the Chat Service

In week 4, we developed the Chat Service, which we will now use as a example of remote hosting.

1. Get the char service server application from LearningSpace/comp260 -week 4.
2. FTP the server application onto your remote server using Filezilla.
3. Edit the server python code to reflect the ip address of the server (46.101.56.200) and the port you are using. Nano is an excellent text editor in linux, to launch just type nano and the name of your file. All the commands are displayed and <ctrl> based. Once you have finished editing your file(s) make sure to save them prior to quitting nano.
4. Open a terminal session in PuTTY
5. Launch your chat server app with nohup. For testing your server, it is worth running it without nohup so that you can read the app's output.
6. In windows, edit your client application to use the correct ip address and port.
7. Run the client app locally, it should connect to your server and work properly
8. Build an exe of you client and share it with your colleagues

Next Steps: Hosting your MUD

Now apply all of this to your MUD application