# Chapter 1 Lab Work: Overview of Distributed Systems

Nguyen Ngoc Lam

Sunday 15th March, 2020

# Contents

# 1 Path of the html File That Contains the Content of the Default Website Apache

You can find it at (on linux): /var/www/html/

# 2 Default Port on Which Webserver Is Listening

The default port is 80. But you can change it in /etc/apache2/ports.conf (remember to switch to superuser first)



Figure 1: Apache port configuration file.

# 3 Meaning of Permission 755

|  | Write | Execute | Read |
|---|---|---|---|
| Value | r | w | x |
| Value (in number) | 4 | 2 | 1 |

Chmod 755 (chmod a+rwx,g-w,o-w) sets permissions so that, (U)ser / owner can read, can write and can execute. (G)roup can read, can't write and can execute. (O)thers can read, can't write and can execute.

# 4 Result of typing 2 addresses

When typing those two addresses, we will have:



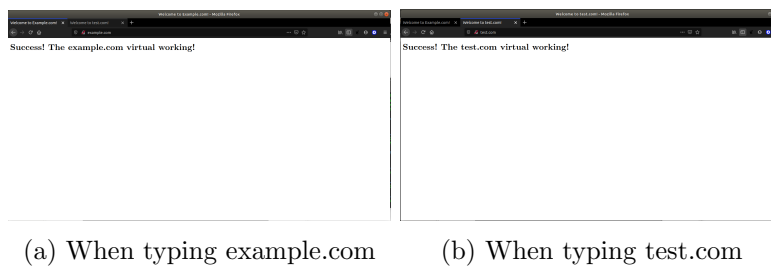(a) When typing example.com        (b) When typing test.com

Figure 2: Result from web browser
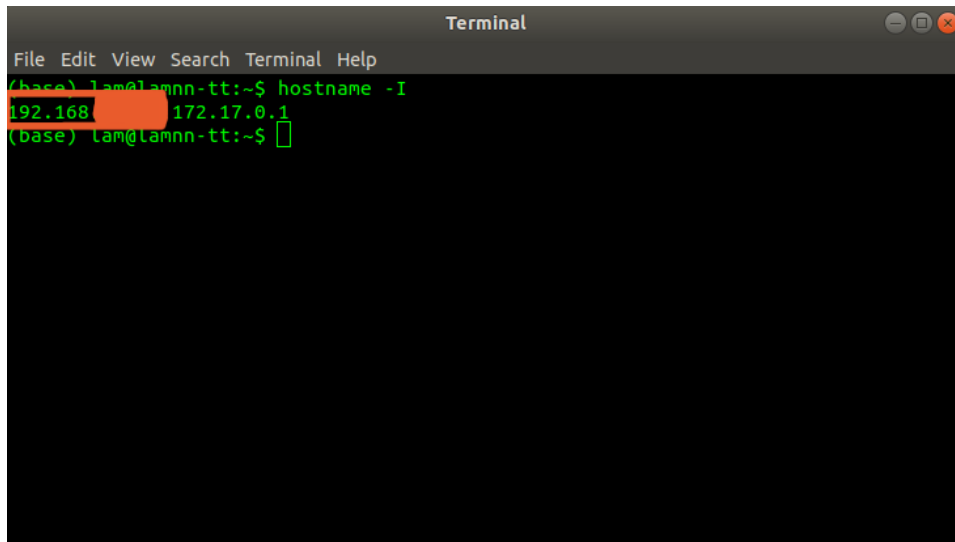
Explain:
When you ran two a2ensite commands for the two conf files, it will enable those two sites, which contained the information stored on the two html files above. After you reload apache to registered those changes and add the ip address (127.0.0.1 or localhost) and map it to the two websites.
The results are showed above: when you type those 2 addresses to browser, it will show the html file.

# 5 Make others machine on LAN to connect to these two websites

1. Type: "hostname -I" onto your terminal, you will see your local IP address (starting with 192.168.) and maybe a broadcast IP address of some other processes (like docker, if you installed) (starting with 172.). We only need the local IP address. We get the result like this:



Figure 3: My local ip address

2. On another computer on the same LAN, open the file /etc/hosts and add these lines with 192.168.x.x is the address you just obtained:
"192.168.x.x example.com"
"192.168.x.x test.com"

# 6    Code for the while loop

```
10 |
11 public class Client {
12⊖     public static void main(String[] args) throws UnknownHostException, IOException {
13          // TODO Auto-generated method stub
14          Socket socket = new Socket("127.0.0.1", 9898);
15          BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
16          PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
17          System.out.println(in.readLine());
18          Scanner scanner = new Scanner(System.in);
19          String message = null;
20          String toserver = "";
21          do {
22              System.out.print("Enter your number here, (leave blank to exit): ");
23              message = scanner.nextLine();
24              toserver = toserver.concat(message);
25              toserver = toserver.concat(" ");
26          } while (!(message.isEmpty()));
27          System.out.println("List confirmed. Sending to server.");
28          out.println(toserver);
29          System.out.println("Sent");
30          System.out.println("Sorted array inbound. It will showed in a moment.");
31          System.out.println("The sorted array:" + in.readLine());
32          socket.close();
33          scanner.close();
34
35      }
```

Figure 4: My client-side

# 7 Role of method run()

- Take the preprocessed input string from client-side.

- Split it in a string array.

- Parse each item in that array into integer, thus make an array of integer.

- Do the sort process (call class)

- Convert the integer array to String

- Send the result to Client and close socket.

```
11
12 public class Server {
13⊖     public static void main(String[] args) throws IOException {
14          // TODO Auto-generated method stub
15          System.out.println("The Sorter Server is running!");      Method run() will be used here
16          int clientNumber = 0;
17          try (ServerSocket listener = new ServerSocket(9898)) {
18              while (true) {
19                  new Sorter(listener.accept(), clientNumber++).start();
20              }
21          }
22      }
23
24⊖     private static class Sorter extends Thread {
25          private Socket socket;
26          private int clientNumber;
27
28⊖         public Sorter(Socket socket, int clientNumber) {
29              this.socket = socket;
30              this.clientNumber = clientNumber;
31              System.out.println("New client #" + clientNumber + " connected at " + socket);
32          }
33
34⊕         public void run() {
73      }
74 }
75
```

Figure 5: My server side

5

# 8   Note

- You can find my poject at: Distributed System repository