BÀI THỰC HÀNH
HỌC PHẦN: HỆ PHÂN TÁN
CHƯƠNG 1: MỞ ĐẦU

1. Web server apache2

1.1. Nội dung

Trên lớp, chúng ta đã được học chương 1: Mở đầu. Chúng ta đã biết rằng tất cả các dịch vụ mạng đều sử dụng lý thuyết của Hệ Phân Tán. Ở bài thực hành này các bạn sẽ thử triển khai dịch vụ WWW. Cụ thể, các bạn sẽ cài đặt 1 web server.

1.2. Yêu cầu

1.2.1. Lý thuyết

- Làm chủ Unix OS
- Kiến thức cơ bản của Mạng máy tính

1.2.2. Thiết bị

- PC hoặc VM

1.2.3. Phần mềm

1.3. Các bước thực hành

1.3.1. Cài đặt web server apache2

Trước hết chúng ta phải cài đặt một phần mềm máy chủ web, và ngày nay thì apache là thông dụng nhất. Hãy sử dụng lệnh sau:

```
sudo apt install apache2
```

Thử truy cập vào webserver này bằng cách gõ địa chỉ IP của mày này từ một máy khác (2 máy cần trong cùng 1 mạng LAN). Các bạn sẽ thấy hiện lên trang web mặc định của apache (Có chữ "Apache2 Ubuntu default page"), có nghĩa là bạn đã cài đặt thành công webserver.

Câu hỏi 1: Đường dẫn đến file html chữa nội dung mặc định của trang web các bạn vừa xem là gì?

Câu hỏi 2: Cổng mặc định của dịch vụ www là gì?

1.3.2. Cài đặt virtual hosts cho apache2

Máy chủ apache2 có khả năng chứa nhiều máy ảo với chỉ 1 địa chỉ IP. Bây giờ chúng ta hãy tạo 2 domains: example.com và test.com. Đầu tiên, tạo 2 thư mục có chứa nội dung cho 2 domains đó:

```
sudo mkdir -p /var/www/example.com/public_html
sudo mkdir -p /var/www/test.com/public_html
```

Thay đổi quyền cho thư mục đó bằng lệnh sau:

```
sudo chmod –R 755 /var/www
```

Câu hỏi 3: Hãy giải thích quyền mang số 755 là gì?

Viết nội dung cho 2 website đó (chỉnh sửa nội dung của file index.html trong 2 thư mục public_html mà bạn vừa tạo ra) như sau:

```
<html>
<head>
<title>Welcome to Example.com!</title>
</head>
<body>
<h1>Success!  The  example.com  virtual  host  is
working!</h1>
</body>
</html>
```

(Nhớ thay đổi thành `test.com` với file tương ứng trong thư mục test.com).

File cấu hình mặc định của các máy ảo của apache là:

```
/etc/apache2/sites-available/000-default.conf
```

Bây giờ hãy tạo 2 file sau:
```
/etc/apache2/sites-available/example.com.conf
/etc/apache2/sites-available/test.com.conf
```

Đây là nội dung của file example.com.conf

```
<VirtualHost *:80>
ServerAdmin admin@example.com
ServerName example.com
ServerAlias www.example.com
DocumentRoot /var/www/example.com/public_html
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Còn đây là nội dung của file test.com.conf :

```
<VirtualHost *:80>
ServerAdmin admin@test.com
ServerName test.com
ServerAlias www.test.com
DocumentRoot /var/www/test.com/public_html
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Chạy 2 lệnh sau để kích hoạt 2 file trên:

```
sudo a2ensite example.com.conf
sudo a2ensite test.com.conf
```

Khởi động lại dịch vụ apache:

```
sudo service apache2 restart
```

Mở file `/etc/hosts` và thêm những dòng sau:

```
127.0.0.1 example.com
127.0.0.1 test.com
```

Bây giờ hãy mở trình duyệt web và thử nghiệm 2 địa chỉ *example.com* và *test.com*

Câu hỏi 4: Bạn quan sát thấy nội dung gì sau khi gõ 2 địa chỉ trên? Giải thích.

Câu hỏi 5: Thử truy cập từ các máy tính khác trong cùng mạng LAN vào 2 trang web đó.

2. Interface trong Java

2.1. Nội dung

Trên lớp, chúng ta đã học về đặc trưng *Tính mở* trong Hệ Phân Tán. Để đảm bảo Tính mở, chúng ta cần phải xây dựng các interface giữa các thành phần của hệ thống. Trong phần thực hành này, chúng ta sẽ xây dựng một mô hình client-server đơn giản, ở đó thì client sẽ gửi một chuỗi số lên cho server. Server sẽ sắp xếp những số nhận được với việc sử dụng phương thức *sort* được khai báo trong interface. Chúng ta sẽ thấy có nhiều cách khác nhau để triển khai phương thức *sort* này (đúng với phát biểu của tính mở là cho phép nhiều nhà sản có thể tham gia vào xây dựng các thành phần theo các cách khác nhau).

2.2. Yêu cầu

2.2.1. Lý thuyết

- Lập trình Java

2.2.2. Thiết bị

- PC

2.2.3. Phần mềm

- Eclipse IDE
- JDK/JRE

2.3. Các bước thực hành

2.3.1. Cài đặt các công cụ cần thiết

- Tải về và cài đặt IDE Eclipse: https://www.eclipse.org/downloads/packages/
- Tải JDK/JRE: https://www.oracle.com/technetwork/java/javase/downloads/index.html

2.3.2. Xây dựng chương trình

Đầu tiên, mở Eclipse và tạo một java project mới với việc chọn *File → New → Java project.* Hãy tự đặt tên cho project đó.
Sau khi tạo xong, bạn sẽ nhìn thấy một thư mục tên là *src*. Thư mục này dùng để chứa mã nguồn.
Tạo 2 packages trong thư mục đó bằng cách ấn phải chuột vào thư mục *src*, sau đó chọn *New → Package*
Đặt tên 2 packages đó như sau:
*com.hust.soict.**your_name**.client_server*
*com.hust.soict.**your_name**.helper*
(trong đó your_name thì thay bằng tên bạn)

Trong package *client_server,* tạo 2 lớp và đặt tên là *Client* và *Server*.
Bây giờ chúng ta sẽ lập trình lên 2 file đó.

2.3.2.1. Client

Hãy mở file Client.java ra và bắt đầu lập trình.
Đầy tiên chúng ta phải import các lớp của bộ thư viện Java:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
```

Sau đó, trong phương thức main, chúng ta sẽ khởi tạo một thực thể socket bằng cách dùng lớp Socket:

```
Socket socket = new Socket("127.0.0.1", 9898);
```

Bạn có thể thay thế địa chỉ IP và số hiệu cổng như bạn muốn, nhưng chú ý đó là địa chỉ IP và cổng mà server sẽ lắng nghe các yêu cầu mà Client sẽ gửi lên.
Bây giờ chúng ta sẽ khởi tạo 2 thực thể của 2 lớp *BufferedReader* và *PrintWriter* để gửi và nhận dữ liệu:

```
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);
```

Khởi tạo 1 thực thể cho lớp Scanner:

```
System.out.println(in.readLine());
Scanner scanner = new Scanner(System.in);
```

Câu hỏi 6: Hãy tự viết một đoạn code để thực hiện 1 vòng lặp while sao cho nó sẽ nhận các số mà người dùng gõ và gửi về server, cho đến khi nào người dùng gõ ký tự rỗng rồi ấn enter. Gợi ý: hãy dùng lệnh sau để nhận xâu ký tự người dùng gõ vào:

```
String message = scanner.nextLine();
```

Cuối cùng, đừng quên đóng các *socket* và *scanner* lại:

```
socket.close();
scanner.close();
```

2.3.2.2. Server

Bây giờ hãy mở và chỉnh sửa file Server.java. Mục đích là xây dựng một server đa luồng để nhận các số mà người dùng gửi lên và sắp xếp nó theo thứ tự tăng dần (hoặc nhỏ dần) và gửi trả lại cho Client.
Đầu tiên hãy import một số thư viện Java cần thiết:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import com.hust.soict.haianh.helper.*;
import java.util.Arrays;
```

Trong phương thức main, hãy viết đoạn code như sau:

```
System.out.println("The Sorter Server is running!");
int clientNumber = 0;
try (ServerSocket listener = new ServerSocket(9898)) {
     while (true) {
        new Sorter(listener.accept(), clientNumber++).start();
     }
}
```

Chú ý là số hiệu cổng phải đúng là số hiệu cổng mà client gửi lên.
Phía ngoài phương thức *main*, hãy tạo 1 lớp *Sorter* được khai báo như một luồng. Và nó phải kế thừa từ lớp Thread:

```
    private static class Sorter extends Thread {
        private Socket socket;
        private int clientNumber;

        public Sorter(Socket socket, int clientNumber) {
            this.socket = socket;
            this.clientNumber = clientNumber;
            System.out.println("New client #" + clientNumber + " connected
at " + socket);
        }
```

```
        public void run() {
            try {
                BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);

                // Send a welcome message to the client.
                out.println("Hello, you are client #" + clientNumber);

                // Get messages from the client, line by line; Each line has
several numbers separated by a space character
                while (true) {
                    String input = in.readLine();
                    if (input == null || input.isEmpty()) {
                        break;
                    }
                    //Put it in a string array
                    String[] nums = input.split(" ");

                    //Convert this string array to an int array
                    int[] intarr = new int[ nums.length ];

                    int i = 0;

                    for ( String textValue : nums ) {
                           intarr[i] = Integer.parseInt( textValue );
                      i++;
                    }

                    //Sort the numbers in this int array
                    new SelectionSort().sort(intarr);
                    //Convert the int array to String
                    String strArray[] = Arrays.stream(intarr)
                                              .mapToObj(String::valueOf)
                                              .toArray(String[]::new);
                    //Send the result to Client
                    out.println(Arrays.toString(strArray));
                }
            } catch (IOException e) {
                System.out.println("Error handling client #" +
clientNumber);
            } finally {
                try { socket.close(); } catch (IOException e) {}
                System.out.println("Connection with client # " +
clientNumber + " closed");
            }
        }
    }
```

Câu hỏi 7: Vai trò của phương thức *run* là gì? Khi nào thì nó được gọi?

Trong đoạn mã trên đây, các bạn có thể thấy phương thức sort được gọi thuộc lớp *SelectionSort*. Bây giờ chúng ta sẽ xây dựng một giao diện (interface) và khai báo phương thức sort ở bên trong. Lớp *SelectionSort* là một trong những lớp triển khai giao diện này.

2.3.2.3. *Interface và các cách triển khai khác nhau*

Bây giờ hãy ấn phải chuột vào package *com.hust.soict.your_name.helper* , chọn *New → Interface*
Tạo một interface mới và đặt tên nó là *NumberSorter*.

Trong file đó, viết đoạn code sau để khai báo phương thức *sort*:

```
public interface NumberSorter {
	void sort(int arr[]);
}
```

Vẫn trong package đó, tạo một lớp khác đặt tên là SelectionSort.
Thực tế là, lớp *SelectionSort* sẽ triển khai giao diện *NumberSorter* và định nghĩa cụ thể phương thức sort dựa trên thuật toán *selection sort*. Mở file SelectionSort.java và viết đoạn code sau:

```
public class SelectionSort implements NumberSorter{
	public void sort(int arr[]) {
		int n = arr.length;

	  // One by one move boundary of unsorted subarray
	  for (int i = 0; i < n-1; i++)
	  {
		// Find the minimum element in unsorted array
		int min_idx = i;
		for (int j = i+1; j < n; j++)
		    if (arr[j] < arr[min_idx])
			min_idx = j;

		// Swap the found minimum element with the first
		// element
		int temp = arr[min_idx];
		arr[min_idx] = arr[i];
		arr[i] = temp;
	  }
	}
}
```

2.3.2.4. Chạy chương trình

Bây giờ hãy chạy thử chương trình. Đừng quên là phải chạy server trước khi chạy client.

2.3.2.5. Triển khai các giải thuật sắp xếp khác

Bây giờ hãy tự mình triển khai các giải thuật sắp xếp khác. Bạn làm y như các bước ở trên (tạo lớp mới trong package helper, sau đó triển khai lại interface NumberSorter). Thử triển khai với 3 giải thuật sau (có thể tham khảo mã nguồn trên Internet):

- Bubble sort
- Insertion sort
- Shell sort