Universidad de Valparaíso
CHILE

Redes de computadores:
Implementación de servidor web iterativo

Facundo Martínez, Valentina Clavijo, Álvaro Pismante, Eduardo Cabrera.

I. Introducción

¿Qué es un servidor web iterativo y cómo poder implementarlo? Éstas son las preguntas que debemos plantearnos para poder comenzar el trabajo, ya que en base a esto tiene que ser posible servir páginas web sencillas. Un servidor web iterativo es aquel que está constantemente esperando un *request* del cliente.

Hay que, además, tener parametrizados en un en un archivo JSON la dirección IP, puerto TCP, directorio raíz donde se ubican las paginas web, etc. El servidor debe estar programado para que el usuario pueda ingresar páginas (las cuales deben estar en el directorio) a través de su navegador y el sistema debe transmitirla. En caso de que ésta no sea encontrada, el servidor deberá transmitir la página notFoundFile.

Se incluirá además la documentación de la arquitectura de su software, que contendrá la Vista Lógica y la Vista de Procesos del modelo 4+1.

II. Preparación

Se pone a disposición una cierta cantidad de códigos, los cuales serán fundamentales para la implementación del servidor.

Dentro de los archivos importantes tenemos los *makefiles*, los cuales se incluyen en la carpeta raíz del proyecto que le dicen a un programa, *make*, que se ejecuta desde la terminal, qué hacer con cada uno de los archivos de código para poder compilarlos.

Junto a el *makefile* "principal" irán 3 carpetas, una de ellas contendrá las páginas simples HTTP que serán utilizadas, otra llevará el notFoundFile (error 404) y la tercera llevará todos los códigos encargados de hacer que todo funcione correctamente.

III. TCPEchoServer

TCPEchoServer es el encargado junto a las liberías de manejar los archivos HTML. Éste es el código que recibe la mayor modificacion en relación al resto, ya que como al encargarse de el manejo de archivos para el correcto funcionamiento de el servidor web.

En el sector /*MODIFICACION*/ se le agregaron las siguientes lineas:

- Sock -> send(**a,b**);

  Ésta función se encarga de enviar información por el socket.
  a representa al estado y **b** el largo de éste.

```
sock ->send("HTTP/1.1 200 OK\r\n",32);
sock ->send("Content-Type: text/html\r\n\r\n",27);
```

- Se definen variables, las cuales representan a el request hecho por el cliente, y las 3 posibles páginas. Éstas son las que mas adelante serán manipuladas en el manejo de archivos.

```
 //lee archivo HTML//
 std::string linea;
 std::string html = "";
 std::string rqe(echoBuffer, 5,12);
 std::string a1 = "pagina1.html";
 std::string a2 = "pagina2.html";
 std::string a3 = "pagina3.html";

 std::ifstream archivo ("../www-data/pagina1.html");
//crear una variable para c/u y compararlla con strings d
 std::ifstream archivo1 ("../www-data/pagina2.html");
 std::ifstream archivo2 ("../www-data/pagina3.html");
 std::ifstream archivoE ("../www-error/error404.html");
```

- Luego se hace todo el análisis de pedido, pues se verá cual página pide. En caso de que la página solicitada no sea reconocida, arrojará hacia la página notFoundFile (error 404).

```
if (rqe==a1){
  if (archivo.is_open()){
    while(getline(archivo,linea)){
     html = html + linea + "\n";
    }
    archivo.close();
  }else{
    std::cout << "no se puede abrir" << "\n";
  }
  sock->send(html.c_str(), html.length());
  delete sock;
}
```

IV. Conclusión

Luego de haber realizado todo el trabajo, se puede comprobar como funciona un servidor web iterativo simple a escala pequeña, con el simple hecho de tener un servidor que retorna un HTML por navegado. Al tener la estructura definida de como funciona el codigo para tener un resultado positivo.