

Sistemas

Ingeniería Multimedia

Distribuidos

middleware

web
servicios





Servicios de Red

Contenido

- práctica 1
- práctica 2
- práctica 3**
- práctica 4

Servicios de Red



Servicios de Red

información y metodología

Contenido

práctica 1
práctica 2
práctica 3
práctica 4



Duración:

- Cinco semanas
(del 26 de febrero al 19 de abril de 2018)



Material necesario:

- PC/Mac/Máquinas virtuales
- Compilador C



Material a conseguir por el estudiante:

- Ejemplos de aplicaciones cliente/servidor en la WWW codificados en C



Grupos:

- Se trabajará con los grupos establecidos en las anteriores prácticas



Servicios de Red

objetivos

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

- ⌚ Aprender a crear servicios distribuidos sobre Internet bajo la arquitectura Cliente/Servidor
- ⌚ Conocer la programación de sistemas distribuidos utilizando mecanismos de comunicación entre procesos en red
- ⌚ Aprender a diseñar un protocolo de aplicación
- ⌚ Conocer la tecnología de programación de red con sockets



Servicio HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

- ④ El objetivo de la práctica es desarrollar un **servidor** y un **cliente web**, i.e., que hablen **HTTP**.
- ④ El servidor se denominará **mi_httpd**, y debe permitir a cualquier usuario, utilizando cualquier navegador web, acceder a las paginas web (**HTML**) y a los recursos asociados que residan en el servidor.
- ④ El cliente se denominará **mi_wget** y deberá permitir acceder a cualquier recurso que proporcione tanto el servidor **mi_httpd**, como cualquier otro servidor web.



Servicio HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

- ④ La aplicación desarrollada deberá implementar, al menos, las siguientes características del protocolo HTTP:
 - Sera un protocolo orientado a conexión, manteniendo la conexión entre cliente y servidor durante todo el proceso de comunicación.
 - Podrá ser un protocolo orientado a texto.
 - Sera un protocolo petición/respuesta. Las peticiones se implementaran mediante órdenes o comandos HTTP que serán contestados por el servidor.
 - Sera un protocolo **sin** estado.
- ④ La implementación se realizará en GNU/Linux con C (utilizando opcionalmente las máquinas virtuales de las practicas anteriores).
- ④ Para la comunicación entre cliente y servidor se utilizará la *librería de sockets* estándar.



Servidor mi_httpd

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

- ④ El servidor se invocará de la siguiente manera:
 - **mi_httpd [parámetros de configuración opcionales]**
- ④ Los recursos del servidor estarán alojados a partir de la carpeta especificada con la directiva **DocumentRoot** (que se podrá obtener de un fichero de configuración denominado **mi_httpd.conf** o de los parámetros pasados al programa cuando se invoca. Por defecto será “./html”)
- ④ Si no se especifica un puerto, se utilizara como puerto por defecto para esta aplicación el (**8080**) o el indicado en el archivo de configuración (directriz **Listen**).
- ④ Abrirá un socket en el puerto indicado y esperará mensajes de un navegador web.
- ④ Debe atender peticiones de forma **concurrente** (peticiones de distintos usuarios de forma simultanea).
- ④ El número de conexiones simultaneas debe ser parametrizable utilizando el archivo de configuración (directriz **MaxClients**).
- ④ Aceptara peticiones HTTP y devolverá respuestas HTTP.
- ④ No es necesario implementar el protocolo HTTP completo, únicamente los comandos que se indican a continuación.



Servidor mi_httpd

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4



Archivo de configuración

- Por defecto será **httpd.conf**
- La sintaxis será:
directiva valor
directiva valor
...
- Directivas obligatorias (no quiere decir que deban estar todas en el archivo):
 - *DocumentRoot*
 - *DirectoryIndex*
 - *MaxClients*
 - *Listen*
- Directivas opcionales:
 - *Las que se estime oportuno*
- Ejemplo de archivo **httpd.conf**:

```
DocumentRoot ./html
DirectoryIndex index.html
MaxClients 10
Listen 8080
```




Ciente mi_wget

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

- ④ El cliente se invocará de la siguiente manera:
 - **mi_wget protocolo dns puerto recurso**
- ④ Si puerto es 0, se utilizará el puerto por defecto para el protocolo indicado.
- ④ Deberá traducir la dirección DNS a IP.
- ④ Abrirá un socket en el **puerto** y dirección IP indicados y enviará una solicitud **GET** sobre el **recurso** solicitado.
- ④ Debe esperar una respuestas HTTP que mostrará por la salida estándar.
- ④ No es necesario implementar el protocolo HTTP completo, únicamente el comando **GET**.
- ④ Se pueden (es conveniente) realizar logs para poder hacer un seguimiento de la aplicación. Los logs deben enviarse a la salida de error estándar.



Notas sobre HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4



Peticiones HTTP

- Cada petición HTTP estará compuesta por tres partes como se muestra en la siguiente tabla (siguiendo el estándar HTTP pero acotando las opciones):

Línea inicial	<p>Define el método HTTP, la url de acceso al recurso y la versión del protocolo HTTP.</p> <p>Un salto de línea indica su fin y donde comienza la sección <i>cabecera</i> del mensaje.</p> <p>Sintaxis: MetodoHTTP+EspacioEnBlanco+URIAccesoRecurso+EspacioEnBlanco+HTTPVersion+SaltoDeLinea</p> <p>Ejemplo: GET /prueba.html HTTP/1.1</p>
Cabecera	<p>Directivas que indican características adicionales de la petición y que pueden influir en el tratamiento de dicha petición por parte del servidor. Cada directiva de cabecera estará separada por un salto de línea. Se debe incluir una línea en blanco antes del cuerpo del mensaje para separarlo del mismo.</p>
Cuerpo	<p>En la práctica, estará vacía (solo tendremos, por tanto, un salto de línea)</p>



Notas sobre HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

Línea inicial

Los métodos que deberá reconocer el servidor son:

GET recurso versión: para devolver el fichero/recurso indicado.

HEAD recurso versión: como la anterior, pero sin incluir el contenido del fichero

PUT recurso versión: para “subir” un fichero al servidor (**opcional**)

DELETE recurso versión: para borrar el fichero del servidor

Aunque existen 4 tipos de URI de acceso a recursos, solo se implementará la más habitual (**absolute path**). En ella se indica la ruta absoluta para localizar el recurso que se obtendrá añadiendo a la URI el contenido de la variable

DocumentRoot.

En caso de no especificarse un recurso concreto, se utilizará **index.html** o el que se indique en la directiva **DirectoryIndex** del archivo de configuración.

Cabecera

Las cabeceras que puede entender un cliente son:

Accept: tipo1, tipo2

Accept-Charset: charset (Ejemplo: UTF-8)

User-Agent: tipo de cliente

host: equipo

Cuerpo del mensaje

Este elemento deberá ignorarse.



Notas sobre HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4



Respuesta HTTP

- Cada respuesta HTTP estará compuesta por tres partes como se muestra en la siguiente tabla (siguiendo el estándar HTTP pero acotando las opciones):

Línea inicial	<p>Indica el estado del servidor como respuesta a la petición realizada.</p> <p>Sintaxis: HTTPVerion+EspacioEnBlanco+CodigoDeEstado+EspacioEnBlanco+DescripcionEstado+SaltoDeLinea</p> <p>Ejemplo: HTTP1.1 200 OK</p> <p>Los códigos de estados pueden ser consultados en: http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6</p>
Cabecera	<p>Directivas que indican características adicionales de la respuesta y que pueden influir en el tratamiento de dicha respuesta por parte del cliente web.</p> <p>Cada directiva de cabecera estará separada por un salto de línea.</p> <p>Se debe incluir una línea en blanco antes del cuerpo del mensaje para separarlo.</p>
Cuerpo	<p>Contenido HTML que será mostrado a los usuarios a través de su cliente web.</p>



Notas sobre HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

■ Línea inicial

- Utilizaremos la versión HTTP1.1.
- Cuando el servidor reciba durante una petición algún método no soportado deberá devolver un código de estado “405” “Method Not Allowed”.
- El servidor debe manejar los siguientes códigos de respuesta:
 - Petición satisfactoria:
 - 200 OK → Para indicar que todo es correcto. Tras línea en blanco debe devolver el recurso solicitado.
 - 201 Created → Se ha subido correctamente el fichero
 - Errores en la petición, especificados mas adelante
 - Errores en el servidor especificados mas adelante.



Notas sobre HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

■ Cabecera

- La sintaxis es:

```
directiva: valor  
directiva: valor  
...
```

- Directivas obligatorias:

- *Connection*
- *Content-Lenght*
- *Content-Type*
- *Server*

- Directivas opcionales:

- *Date: fecha*
- *Last-Modified: fecha*
- *Cache-control: no-cache*
- *Cache-control: max-age = seconds*



Notas sobre HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

■ Cuerpo

- Contendrá el código HTML contenido en el fichero referenciado en la petición.
- El servidor no formatea, adapta ni transforma ningún contenido, solo sirve los recursos o página de error en caso de no poder hacerlo.
- Ante cualquier duda se deberá seguir la especificación estándar de HTTP

<http://www.w3.org/Protocols/rfc2616/rfc2616.html>



Notas sobre HTTP

especificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

■ Errores típicos

- Los errores que debe detectar y manejar adecuadamente (es decir, que debe devolver un mensaje que indique cada uno de ellos) el servidor son los siguientes:
 - 400 bad request → Petición errónea
 - 403 forbidden → Petición denegada
 - 404 not found
 - 405 method not allowed
- Errores en el servidor:
 - 500 Internal Server Error
 - 503 Service Unavailable
 - 505 HTTP version not Supported



Servicio HTTP

planificación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

- Ⓢ **Fase 1.** Programar la aplicación **mi_wget** y verificarla adecuadamente
- Ⓢ **Fase 2.** Programar la versión 1 de **mi_httpd**
 - Debe ser capaz de atender una solicitud **GET** en el puerto indicado. No tendrá que leer configuración alguna ni de la línea de comandos ni del archivo de configuración.
- Ⓢ **Fase 3.** Programar la versión 2 de **mi_httpd**
 - Actúa como la versión 1 pero se podrá parametrizar tanto mediante las opciones de la línea de comando como mediante el archivo de configuración.
- Ⓢ **Fase 4 y posteriores (versión 3.x de mi_httpd)**
 - El servidor comenzará a atender el resto de comandos HTTP.
- Ⓢ Utilizar siempre **mi_wget** y otro **navegador web estándar** para verificar el funcionamiento de las diferentes versiones de **mi_httpd**
- Ⓢ Utilizar **Postman** o similar para verificar la fase 4.



Servicio HTTP

entrega

Contenido

práctica 1
práctica 2
práctica 3
práctica 4

- Se deben entregar los fuentes de la aplicación organizados en las diferentes versiones expuestas
- Entregar un breve informe sobre la aplicación, características básicas implementadas, características opcionales y características adicionales que los estudiantes hayan considerado
- La entrega se realizará a través de UACloud
- La entrega será individual (aunque el material puede ser el mismo para los miembros de un mismo grupo)
- El tamaño máximo del informe es de 8 páginas en formato PDF
- La fecha tope de entrega será el 25 de abril



Servicio HTTP

evaluación

Contenido

práctica 1
práctica 2
práctica 3
práctica 4



Clientes Web (hasta 2 puntos)

- Funciona con el cliente **mi_wget**
- Acepta peticiones y las muestra correctamente
- Funciona con un navegador
- Acepta peticiones y las muestra correctamente



Servidor Web *Mi_httpd* (hasta 7 puntos)

- Acepta el comando HTTP **GET**
- Acepta el comando HTTP **HEAD**
- Acepta el comando HTTP **PUT**
- Acepta el comando HTTP **DELETE**
- Si se solicita un recurso inexistente se controla el error
- Se gestionan el resto de errores y códigos de respuesta
- Utiliza todas las cabeceras (respuesta) indicadas
- Utiliza todas las cabeceras (de petición cliente) indicadas
- Se gestionan el resto de errores en el lado del SERVIDOR
- Si se solicita un recurso estático (HTML) existente se sirve adecuadamente
- Se utiliza un fichero de configuración (**httpd.conf**)
- Se ha parametrizado el puerto (**Listen**)
- Se ha parametrizado el **DocumentRoot**
- Se ha parametrizado el recurso por defecto (**DirectoryIndex**)
- Se ha parametrizado el número máximo de conexiones (**MaxClients**)

Contenido

práctica 1
práctica 2
práctica 3
práctica 4



UNIX. Programación avanzada

- *Autor:* Fco. Márquez García
- *Editorial:* ra-ma



The Definitive Guide to **Linux** Network Programming

- *Autor:* Keir Davis, John W. Turner y Nathan Yocom
- *Editorial:* Apress

Sistemas

Ingeniería Multimedia

Distribuidos

middleware

web
servicios