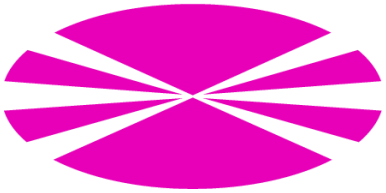


UNIVERSIDADE DA CORUÑA
Departamento de Tecnoloxías da Información
e as Comunicacións

LABORATORIO DE RC:

PRÁCTICA 1: IMPLEMENTACIÓN DE UN SERVIDOR WEB

PRÁCTICA 3: Implementación de un Servidor Web

Esta práctica consiste en el desarrollo de un servidor Web multi-thread, capaz de procesar múltiples solicitudes de servicio simultáneas en paralelo. El objetivo final será obtener un servidor Web capaz de interactuar con un cliente para navegar a través de un sitio Web.

Para la realización del servidor Web se utilizará el protocolo HTTP versión 1.0 (definido en el RFC 1945), en donde se generan solicitudes HTTP independientes para cada componente de la página Web. El servidor procesará múltiples solicitudes en paralelo, utilizando para ello múltiples hilos de ejecución.

En el hilo principal, el servidor permanece escuchando en un puerto fijo y cuando se recibe una petición de conexión TCP, se establece la conexión a través de otro puerto y se resuelve el servicio en un hilo de ejecución separado.

Para la realización de la práctica se recomienda una aproximación en dos etapas. En una primera etapa, el servidor multi-thread únicamente mostrará por pantalla el contenido de las solicitudes HTTP recibidos, para a continuación (una vez que el programa se ejecuta convenientemente) incluir el código que genere la respuesta adecuada. Para comprobar el correcto funcionamiento del servidor se recomienda utilizar el comando `nc` para conectarse al puerto de escucha, e introducir directamente por línea de comandos las órdenes HTTP.

El servidor deberá lanzarse en un puerto no reservado (i.e. por encima del puerto 1024), y en el cliente se le deberá indicar el fichero que se está solicitando. A modo de ejemplo, una URL que se podría solicitar desde el navegador tendría el siguiente formato: http://localhost:1111/index.html. El servidor Web localizará el fichero solicitado y será devuelto al cliente en el formato adecuado. En caso de que se produzca algún error (por ejemplo, el fichero solicitado no se encuentra), el servidor deberá responder con el código HTML adecuado para cada situación errónea.

La versión básica de esta práctica debe incluir:

- Implementación de un servidor Web multi-thread.
- Implementación de los métodos GET y HEAD.
- Soporte de los formatos básicos HTML, texto plano, GIF y JPEG.

Además, dentro de la práctica se incluyen las siguientes opciones:

- Implementación de la opción “If-modified-since” para el método GET, que se deberá activar a través del navegador Web mediante la recarga de la página (botón “Actualizar”). De esta manera, el navegador solicitará al servidor Web la página Web con la opción “If-modified-since”, descargándola si ha habido cambios. Para verificar el correcto

funcionamiento de esta opción se recomienda utilizar el complemento "*Tamper Data*" de Firefox.

- Definición de dos ficheros de log: para accesos y para errores.
    - El fichero de log para accesos tendrá el siguiente formato:
        - Dirección IP del cliente que realiza la solicitud.
        - Fecha y hora en la que se recibió la petición: [día/mes/año hora:minuto:segundo zona_horaria]
        - Línea de petición del cliente (entre comillas dobles).
        - Código de estado que el servidor envía como respuesta al cliente.
        - Tamaño (en bytes) del objeto enviado al cliente.
    - El fichero de log para errores tendrá el siguiente formato:
        - Fecha y hora en la que se produjo el error (mismo formato que para el fichero de log de accesos).
        - Dirección IP del cliente que generó el error.
        - Mensaje de error.
        - Petición que provocó el error.
- Configuración básica del servidor Web: el servidor Web dispondrá de un **fichero de configuración** a través del cual se podrán indicar, al menos, los siguientes parámetros (se recomienda utilizar la clase java.util.Properties para el desarrollo de esta opción):
    - Parámetro PORT: Puerto de inicio
    - Parámetro DIRECTORY_INDEX: Nombre del fichero a utilizar por defecto, en caso de no incluirse en la petición (por ejemplo, http://localhost:1111/)
    - Parámetro DIRECTORY: Directorio raíz del servidor Web, en donde se ubicarán las páginas Web. Se incluirá la directiva ALLOW, que funcionará de la siguiente manera:
        - Si existe el fichero por defecto, se mostrará el fichero por defecto.
        - Si está activada y no existe fichero por defecto en ese directorio, se mostrará el listado de todos los archivos/directorios que lo componen, con un enlace que permitirá abrir cada fichero/directorio.
        - Si no está activada y no existe el fichero por defecto, se mostrará el error 403 (Access forbidden).

- Ejecución de CGIs (*Common Gateway Interface*): el servidor Web deberá crear un subproceso y ejecutar un programa especificado en la petición, recibiendo como parámetros de entrada los datos enviados por el navegador a través del método GET. Será necesario capturar la salida estándar del proceso y el resultado será enviado directamente al navegador. Para comprobar el correcto funcionamiento de esta opción se utilizará un **formulario** cuyos datos serán procesados por un CGI que mostrará el resultado por pantalla. Para esta opción se dispondrá de un saludo.html y de un saludoCGI.java, si bien también se puede optar por utilizar un formulario y un CGI propios.

Será necesario definir un directorio específico para la localización de los CGIs con permisos de ejecución.

Además, la práctica deberá ser realizada utilizando las funcionalidades propias de los sockets en Java, sin utilizar ninguna clase que implemente total o parcialmente las características del protocolo HTTP (como por ejemplo, la clase HttpURLConnection).

Introducción a HTTP

El protocolo HTTP (HyperText Transfer Protocol) está especificado en el RFC 1945 (versión 1.0) y en el RFC 2616 (versión 1.1). Este protocolo define cómo los clientes (navegadores) solicitan páginas Web a los servidores Web, y cómo éstos realizan la transferencia de estas páginas.

El protocolo HTTP se basa en el protocolo TCP (que ofrece un servicio orientado a conexión y fiable), lo que garantiza que cada mensaje HTTP emitido por el cliente o el servidor es recibido en el otro extremo sin sufrir modificaciones. Además, HTTP es un protocolo sin estado, esto es, el servidor HTTP no almacena ningún tipo de información sobre sus clientes. Cada petición recibida por el servidor se trata independientemente de las peticiones anteriormente recibidas de ese u otros clientes.

El protocolo HTTP 1.0 utiliza conexiones no persistentes, ya que es necesario establecer una conexión TCP independiente para cada uno de los componentes de una página Web. De manera esquemática, el procedimiento para la petición de una URL (p.e. http://www.tic.udc.es/index.html) sería el siguiente:

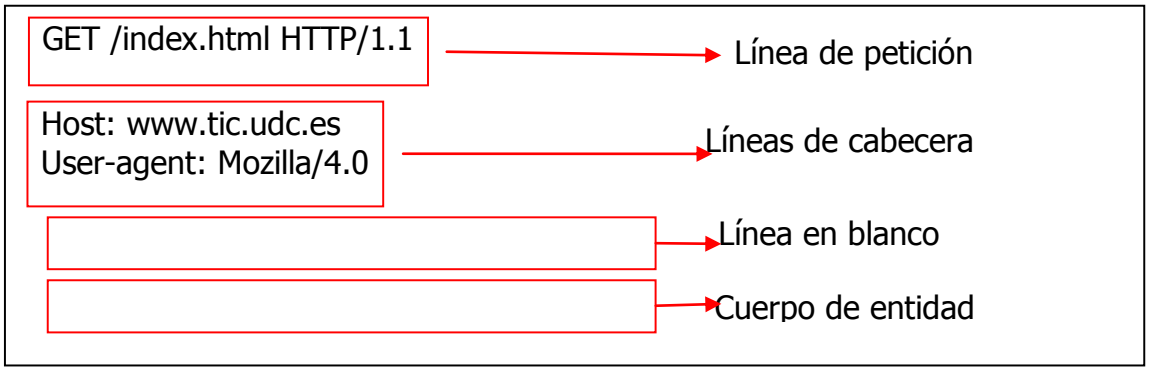
1. El cliente HTTP inicia la conexión TCP con el servidor www.tic.udc.es en el puerto 80.
2. El cliente HTTP envía al servidor el mensaje de petición solicitando el objeto /index.html
3. El servidor HTTP recibe la petición, lee el objeto, lo encapsula en el mensaje HTTP de respuesta y lo envía.
4. El servidor finaliza la conexión TCP.

5. El cliente HTTP recibe la respuesta y finaliza la conexión TCP.
6. El cliente extrae el archivo del mensaje de respuesta, examina el archivo HTML y encuentra referencias a otros objetos HTML (p.e. imágenes)
7. Volver al paso 1, para cada uno de los nuevos objetos HTML.

El principal inconveniente de esta aproximación se centra en el retardo introducido, ya que es necesario esperar dos veces el RTT (Round Trip Time), uno para el establecimiento de la conexión y otro para la petición y respuesta del objeto solicitado. Además, el hecho de utilizar una conexión para cada objeto solicitado implica una mayor utilización de recursos (buffers, variables, timeouts, …), tanto en el cliente como en el servidor.

Estos problemas se resuelven mediante el protocolo HTTP 1.1 que utiliza conexiones persistentes, en donde el servidor deja abierta las conexiones TCP establecidas en espera de nuevas peticiones. Tras un período de inactividad estas conexiones se cierran por parte del servidor.

El protocolo HTTP sigue un sencillo modelo de peticiones y respuestas. El formato de una petición se puede observar en la siguiente figura. Los únicos campos obligatorios son la línea de petición y la línea en blanco.



La línea de petición especifica el tipo de petición que se está realizando, y está formada por tres campos:

- Método
- URL: objeto al que se hace referencia.
- Versión: del protocolo HTTP utilizada por el navegador.

Los métodos básicos definidos en el protocolo HTTP son los siguientes:

- GET: solicitud de un objeto por parte del cliente.
- HEAD: se solicita al servidor que responda con un mensaje HTTP, aunque sin incluir el objeto en la respuesta.
- POST: permite incluir datos en el cuerpo de entidad.
- PUT: permite a un cliente cargar un archivo en la ruta especificada (sólo en HTTP 1.1).

- DELETE: permite a un cliente borrar un archivo de un servidor (sólo en HTTP 1.1).

Dentro las líneas de cabecera, un navegador típico puede incluir múltiples opciones, siendo las más relevantes:

- Host: especifica el host en el que reside el objeto solicitado.
- User-agent: especifica el tipo de navegador que está haciendo la petición.
- If-modified-since: utilizada junto con el método GET, especifica una fecha para que el servidor, si el objeto no ha sido modificado con posterioridad a esa fecha, no lo envíe de nuevo.

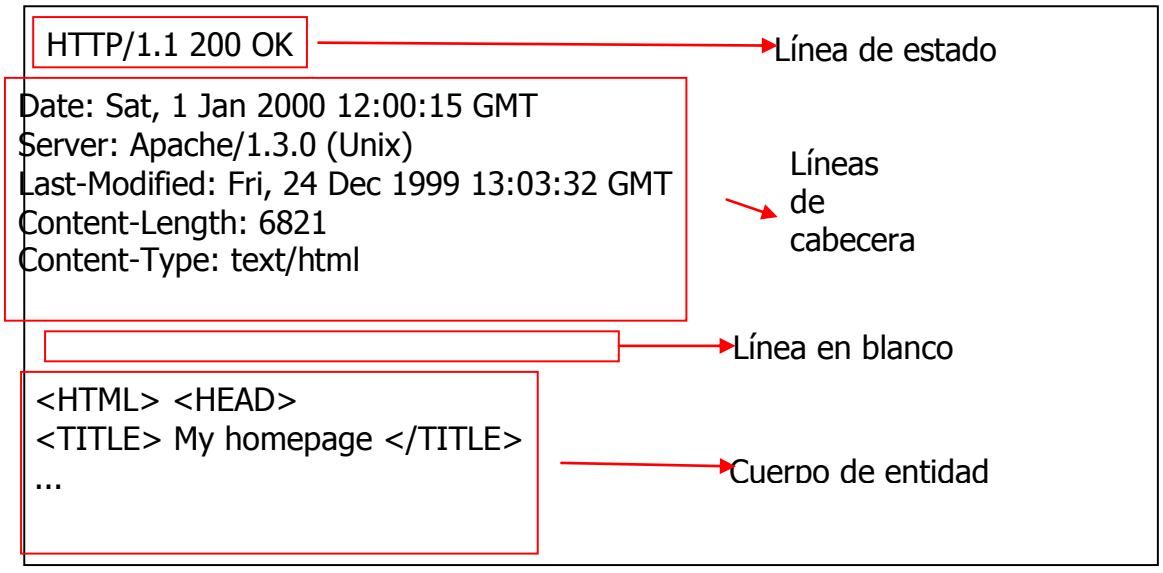
La opción "*If-modified-since*" es utilizada en las implementaciones de las cachés en el cliente. El formato de la petición sería el siguiente:

```
GET /images/udc.gif HTTP/1.1
User-agent: Mozilla/4.0
If-modified-since: Fri, 24 Dec 1999 13:03:32 GMT
```

Mientras que en el servidor, si el objeto no ha sido modificado, la respuesta sería la siguiente:

```
HTTP/1.1 304 Not Modified
Date: Wed, 5 Jan 2000 20:30:43 GMT
Server: Apache/1.3.0 (Unix)
```

El formato de las respuestas HTTP se puede observar en la siguiente figura.



La línea de estado incluye tres campos:

- Versión: versión utilizada por el servidor Web.
- Código de estado: código numérico que representa si la respuesta ante la petición es satisfactoria o si ha habido algún error.
- Frase: asociado a cada código numérico existe una frase que informa sobre la naturaleza del código. En el RFC 1945 se pueden consultar todos los códigos y frases del protocolo, destacando los siguientes:
    - 200 OK
    - 400 Bad Request: petición no comprendida por el servidor.
    - 403 Forbidden: petición comprendida por el servidor, pero que rechaza satisfacer.
    - 404 Not Found: el objeto solicitado no existe en el servidor.

Respecto a las líneas de cabecera, existen múltiples parámetros que el servidor puede especificar, destacando los siguientes:

- Date: fecha y hora en la que se creó y envió la respuesta HTTP.
- Server: especifica el tipo de servidor Web que ha atendido la petición.
- Last-Modified: indica la fecha y hora en que el objeto fue creado o modificado por última vez.
- Content-Length: indica el número de bytes del objeto enviado.
- Content-Type: indica el tipo de objeto incluido en el cuerpo de entidad. Este campo es necesario, ya que la extensión del archivo no especifica (formalmente) el tipo de objeto asociado. Los tipos más comúnmente utilizados son:
    - text/html: indica que la respuesta está en formato HTML.
    - text/plain: indica que la respuesta está en texto plano.
    - image/gif: indica que se trata de una imagen en formato gif.
    - image/jpeg: indica que se trata de una imagen en formato jpeg.
    - application/octet-stream: utilizado cuando no se identifica el formato del archivo.

FECHA DE ENTREGA

La práctica 3 se iniciará la semana del 18 al 22 de Octubre de 2010. La defensa de la práctica se realizará, **como máximo**, la semana del 15 al 19 de Noviembre de 2010. Para ello, cada alumno estará apuntado en uno de los grupos de prácticas de la asignatura y la práctica será presentada y defendida en el día de

prácticas habitual. En caso de hacerlo en un grupo posterior se podrá considerar que la práctica ha sido entregada fuera de plazo.

Para la presentación de esta práctica, se mostrará su funcionamiento en el laboratorio de prácticas y el alumno deberá ser capaz de explicar el funcionamiento de cualquier componente de la misma.

Además, el código fuente de la práctica deberá ser copiado al repositorio de entrega de prácticas una vez finalizada su defensa. En caso contrario, se considerará que la práctica NO ha sido presentada.