1 Sesión 1. Protocolo HTTP.

En esta práctica se va a incidir en varios aspectos del protocolo HTTP que junto a las URIs y el HTML establecen la semántica de la Web.

1.1 Métodos GET y POST

El método `GET` permite solicitar un recurso multimedia (texto, imagen, sonido, PDF, etc.) a un servidor Web. También permite enviar datos al servidor en la URL que identifica el recurso.

El método `POST` permite enviar datos al servidor Web para su procesado. Es similar al `GET`, pero además envía datos en el cuerpo del mensaje y, en este caso, la URL identifica una página web dinámica.

La sintaxis de una petición `GET` o `POST` es la indicada en la Figura 1.

Figura 1. Sintaxis de una petición

dónde se ha utilizado `<CRLF>` (*Carriage Return / Line Feed*) para indicar que entre las cabeceras y el cuerpo del mensaje debe de haber una línea en blanco.

Las cabeceras son pares propiedad/valor separados por el carácter '`:`'. El cuerpo del mensaje es vacío si el método es `GET` (recuérdese que de enviarse datos al servidor éstos irían en la URL) y constaría de los datos que se envían al servidor (codificados) si el método es `POST`.

Después de recibir e interpretar una petición, `GET` o `POST`, el servidor retorna al cliente una respuesta con el formato indicado en la Figura 2.

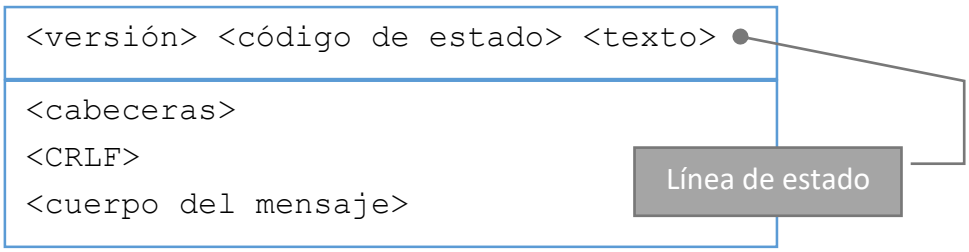

Figura 2. Mensaje de respuesta del servidor

1.1.1 Ejemplo del método GET

Para ver el formato de una petición `GET` correspondiente a la solicitud de un recurso utilizaremos en primer lugar la extensión *Live HTTP Headers* para el navegador Chrome. Abre el navegador Chrome e instala la extensión mencionada.

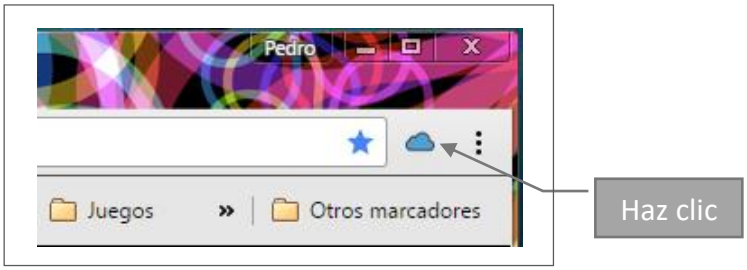

Figura 3. Icono de la extensión Live HTTP Headers

Una vez instalada la extensión e invocada ésta haciendo clic en el icono correspondiente (ver Figura 3), se abrirá una nueva pestaña en el navegador (ver Figura 4) dónde se mostrarán las peticiones que el cliente haga a los servidores Web. Por defecto, ya está habilitado el modo *capture* e indicado que muestre los mensajes en modo *raw*; es decir, como texto y casi exactamente como se indicó en el apartado 1.1. Debes tener en cuenta que en esta utilidad nunca se mostrará el cuerpo del mensaje, sólo las líneas de petición o estado y las cabeceras. Si el mensaje es una respuesta del servidor a una petición, el cuerpo de éste se interpretará y mostrará en la ventana del navegador en la que se realizó la petición y si corresponde a datos de una petición, éstos se enviarán al servidor.

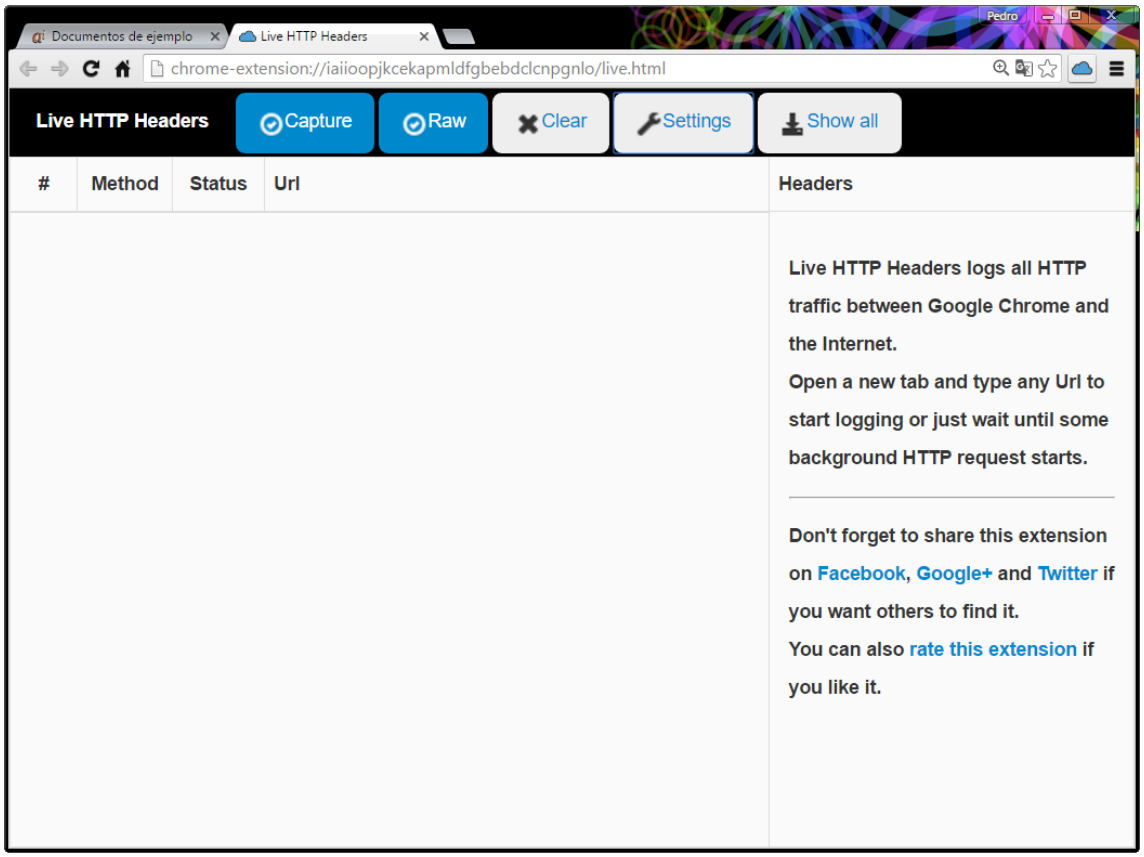

Figura 4. Vista de la extensión Live HTTP Headers

Cierra todas las pestañas del navegador a excepción de la correspondiente a la ventana de la utilidad. Abre ahora otra pestaña del navegador para solicitar el recurso identificado mediante la siguiente URL:

http://www-sew.dyndns.org/sew/practicas/practica-01/w3c.png

Tras solicitar el recurso se mostrará una imagen y en la ventana de la utilidad se habrán capturado las peticiones `GET` realizadas por el cliente. Observarás que no sólo se ha solicitado el recurso identificado por la URL indicada, también se ha solicitado el recurso identificado mediante la URL:

http://www-sew.dyndns.org/favicon.ico

esto es algo que suele ser habitual en los navegadores, pero no en otros clientes Web. Esta última URL identifica un recurso que, supuestamente, corresponde al icono del logotipo de la entidad responsable del sitio Web, icono que el navegador muestra en la pestaña de la ventana en la que se solicitó algún recurso de dicho sitio.

Por último, si en la ventana de la utilidad haces clic sobre la URL inicial, se mostrará el mensaje correspondiente a la petición del recurso y el mensaje correspondiente a la respuesta del servidor Web (sin el cuerpo en ambos casos).

Fíjate bien en el formato de ambos mensajes, inicialmente en las líneas de petición y de estado. En ambas, la versión del protocolo es el texto `HTTP/1.1` o podría ser `HTTP/1.0`, dado que la versión inicial del protocolo es la `0.9`, ¿también podría aparecer la versión cómo `HTTP/0.9`? Piensa la respuesta.

Observa que en la línea de petición no figura la URL completa del recurso, sólo la *ruta* (o *camino*) de éste. Recuerda que el navegador descompone la URL en tres partes:

1. El protocolo (`http`)
2. El nombre del servidor (`www-sew.dyndns.org`), que podría ser directamente su dirección IP (`156.35.163.103`).
3. La ruta o camino (`/sew/practicas/practica-01/w3c.png`)

y que el nombre del servidor (o su IP) se utiliza como valor de la cabecera `Host` en la petición.

Puedes probar ahora a desactivar el modo *raw* de la utilidad haciendo clic en el botón correspondiente. Al desactivar el modo *raw* se separan las líneas de petición y estado, que se presentan en primer lugar con un fondo de distinto color, y se formatean las cabeceras para hacerlas más legibles (ya no estás viendo los mensajes de petición y respuesta del protocolo HTTP tal y como viajan realmente).

Ahora fíjate en las cabeceras (pares propiedad/valor), comenzando por las cabeceras de la petición. Éstas informan al servidor de: los contenidos que acepta el cliente y la calidad con que los trata (que por defecto es 1), si admite mensajes comprimidos y en que formato, los lenguajes que acepta, el sistema operativo del cliente y el tipo de cliente utilizado.

Respecto a las cabeceras de respuesta del servidor, éstas informan al cliente de: el tamaño y tipo MIME del cuerpo del mensaje (las dos cabeceras más importantes del mensaje de respuesta, sobre todo la segunda), la fecha en que se originó el mensaje, una etiqueta que identifica el recurso y que no cambia en tanto éste no se modifique en el servidor (esto permite que un cliente pueda recuperar recursos de su caché mediante peticiones condicionales), la fecha de la última modificación y el tipo de servidor Web utilizado y sistema operativo de la máquina en que reside.

Recuerda que todo el mensaje HTTP es texto, de tal forma que para el que el cliente (o el servidor si se envían datos `POST` a éste) sepa cómo debe interpretar y tratar el cuerpo del mensaje es imprescindible la etiqueta `Content-Type` que establece su tipo MIME.

Si tienes dudas, o quieres saber más sobre las cabeceras de los mensajes HTTP, consulta en línea la sección correspondiente de la RFC 2616:

https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

1.1.2 Ejemplo del método POST

Veremos ahora un ejemplo del método `POST`. En primer lugar, acuérdate de activar el modo *raw* de la utilidad si lo habías desactivado y luego, en la otra ventana, cubre el formulario que se presenta al acceder a la página web dinámica identificada mediante la URL:

http://www-sew.dyndns.org/sew/practicas/practica-01/ejemplo.php

Tras enviar los datos del formulario (al pulsar el botón correspondiente) repite los pasos realizados en el apartado previo. Al igual que antes podrás visualizar la línea de petición o de estado y las cabeceras de los mensajes de petición y respuesta, pero no se mostrará el cuerpo de éstos.

EJERCICIO 1

Abre un navegador y borra los datos de éste (fundamentalmente la información de su caché). Solicita ahora al recurso identificado mediante la URL:

http://www-sew.dyndns.org/sew/practicas/practica-01/ejemplo.html

y tras ver la respuesta que se muestra en el navegador y teniendo en cuenta que la página a la que se accede carece de *scripts* y de información de estilo (hojas CSS), trata de responder a la siguiente pregunta ¿cuántas peticiones `GET` se hacen al servidor Web para obtener el resultado que se muestra en el navegador?

Nota: Deberías ser capaz de responder a la pregunta sin necesidad de utilizar utilidad o herramienta alguna. Si no fuera el caso (o comprobar que estás en lo cierto), utiliza la extensión *Live HTTP Headers* para obtener la respuesta a la cuestión planteada y piensa por qué es ese el número de peticiones realizadas. Si aun así no tienes claro por qué es esa la respuesta correcta, pregunta.

1.2 Uso de la herramienta netcat

Para poder mostrar todo el formato de los mensajes, incluido su cuerpo, utilizaremos la herramienta `netcat` (`nc.exe`). Esta herramienta se ejecuta desde el intérprete de

comandos (*shell*) y permite abrir puertos TCP/UDP en un anfitrión (*host*) quedando a la escucha y volcando a la salida estándar todo lo que le llega. Su nombre está derivado del comando `cat` de Unix, este comando vuelca a la salida estándar los caracteres de un archivo y el `netcat` básicamente hace lo mismo, la única diferencia es que en éste caso lo que se vuelca a la salida estándar son todos los caracteres que llegan al puerto (*socket*) de la conexión TCP/UDP (TCP por defecto) establecida con el anfitrión.

1.2.1 Netcat en modo cliente

La herramienta `netcat` está disponible en la carpeta `netcat-1.11` contenida en el archivo comprimido `practica-01.zip` que se encuentra en el campus virtual. Descomprime el contenido de este archivo en la carpeta `practica-01` del `Escritorio` (o de la carpeta `Mis Documentos`).

La sintaxis y modo de trabajo del comando `netcat` figura en el `readme.txt` que está incluido en la carpeta `netcat-1.11` de la práctica y según los parámetros que se proporcionen puede actuar como *cliente* o como *servidor*. También puedes utilizar la Wikipedia para ver los parámetros y algunos ejemplos de uso (aunque éstos están enfocados al *shell* del Unix, los más simples se pueden utilizar igualmente en el *shell* de Windows).

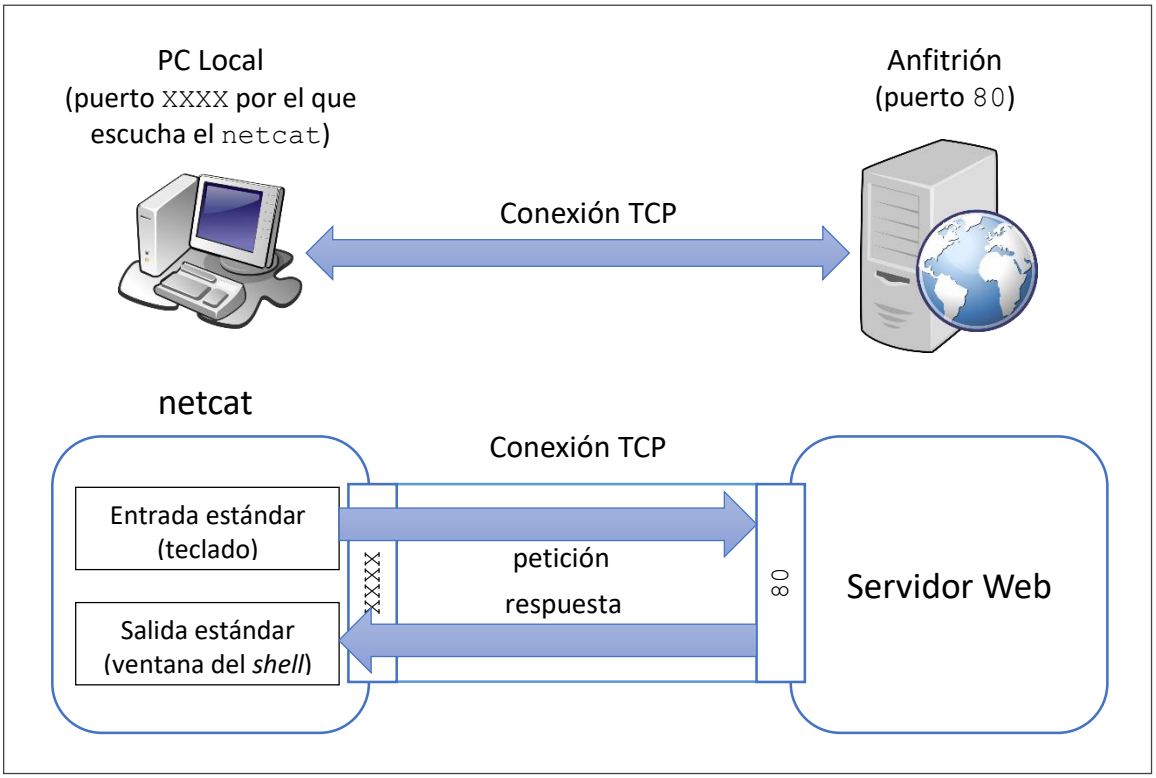

Figura 5. Conexión del netcat a un servidor Web

En primer lugar, se establecerá una conexión TCP con el servidor Web del anfitrión `www-sew.dyndns.org` que está a la escucha por el puerto `80` (puerto por defecto); es decir, en este caso el `netcat` actuará como cliente y escuchará el puerto abierto (`XXXX`) en tú PC local por la conexión TCP establecida con el puerto `80` del anfitrión (ver Figura 5). Por tanto, lo único que podrá llegar al puerto de tú PC, y volcarse a la salida estándar

(la ventana del *shell*), es una respuesta del servidor a una petición previa del cliente. Y para realizar ésta hay que escribir el mensaje de solicitud de un recurso del sitio Web (con la sintaxis requerida) en el puerto `XXXX` por medio de la entrada estándar (el teclado). Este modo de funcionamiento es el que se ha reflejado de forma esquemática en la Figura 5.

El comando para abrir una conexión TCP entre el `netcat` y el anfitrión es muy simple, sólo se requieren dos parámetros: el nombre del anfitrión (o su dirección IP) y el número de puerto del anfitrión al que queremos conectarnos. Sin embargo, de esta forma estaríamos obligados a teclear el mensaje de solicitud del recurso del servidor Web, lo que resulta tedioso, puede dar lugar a errores sintácticos en el mensaje con suma facilidad y probablemente lleve a superar el tiempo de desconexión por inactividad (*timeout*) antes de completar el mensaje a enviar.

La alternativa a utilizar directamente el teclado para realizar la petición, es utilizar un archivo cuyo contenido sea el texto del mensaje a enviar junto con la facilidad que proporciona normalmente un intérprete de comandos para redirigir tanto la entrada como la salida a un archivo. En el caso de la entrada leyendo de éste y en el de la salida escribiendo en él.

Utiliza ahora el editor `Notepad++` para crear un archivo cuyo contenido sea exactamente la secuencia de caracteres que enviaría un navegador (u otro cliente Web) al servidor para solicitar el recurso identificado por la URL:

http://www-sew.dyndns.org/sew/practicas/practica-01/ejemplo.html

Una posibilidad sería copiar y pegar el formato del mensaje tal cual se obtiene en el modo *raw* con la utilidad *Live HTTP Headers*, pero esto sería contraproducente por contener ciertas cabeceras que harían creer al servidor que el cliente dispone de ciertas facilidades de las cuales el `netcat` en realidad carece. Es suficiente con que incluyas las tres líneas que se indican a continuación: *la línea de petición*, *la cabecera* `Host` (obligatoria en HTTP/1.1) y *la línea en blanco* que separa las cabeceras del cuerpo del mensaje (que en este caso estará vacío). Llama al archivo `peticionGET.txt` y guárdalo en la carpeta `practica-01` que creaste para descomprimir el archivo `practica-01.zip`.

Líneas de `peticionGET.txt`:

```
1 GET /sew/practicas/practica-01/ejemplo.html HTTP/1.1
2 Host: www-sew.dyndns.org
3
```

Antes de guardar el archivo asegúrate de ubicar el cursor al principio de una cuarta línea, en caso contrario la marca `EOF` de fin de archivo iría en la tercera línea y ya no habría separación entre las cabeceras y el cuerpo del mensaje (vacío), produciéndose un error: *el servidor no enviaría mensaje de respuesta alguno por qué la petición estaría incompleta*.

Ahora lanza el intérprete de comandos, pulsa las teclas `Windows + R` para abrir el diálogo `Ejecutar` y escribe la orden que abre el *shell*: `cmd.exe`. En la ventana del

intérprete de comandos utiliza la orden `cd` para ubicarte en el directorio `practica-01` y ejecuta la orden `nc.exe` (`netcat`) proporcionando todos los parámetros necesarios, incluida la redirección de la entrada estándar, en la misma línea:

```
                Orden netcat          Host        Puerto  Redirección
$ netcat-1.11\nc www-sew.dyndns.org 80 <peticionGET.txt
```

Ten en cuenta que no es posible ejecutar el comando `nc` de ninguna otra forma para redirigir la entrada estándar, así que no hagas doble clic sobre el archivo `nc.exe`, ni trates de poner los mismos parámetros si estos son solicitados por la orden `nc` (ver Figura 6).

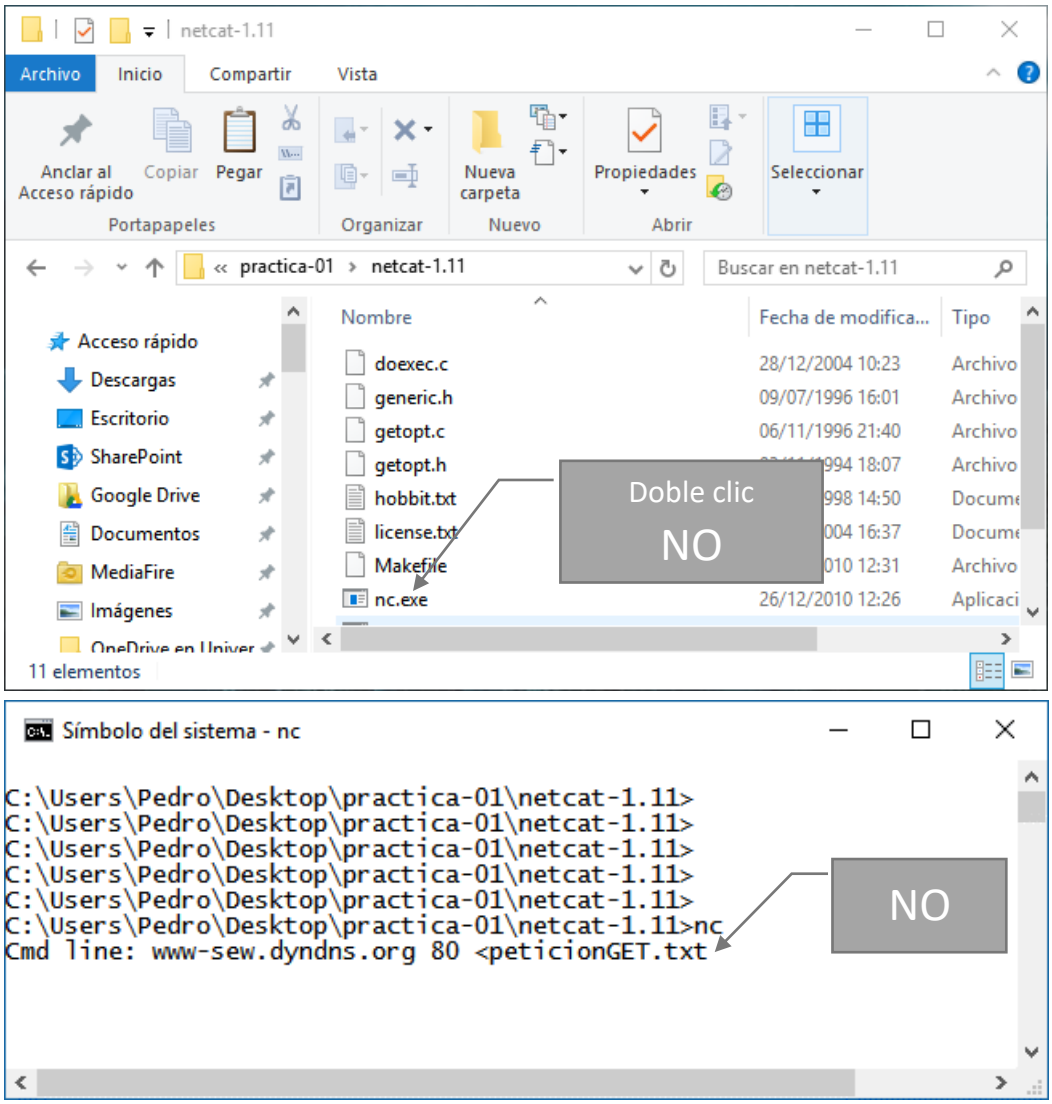

Figura 6. Formas de ejecución del netcat que no deben utilizarse

EJERCICIO 2

Haz una copia del archivo `peticionGET.txt` que creaste en la carpeta `practica-01` en la misma carpeta y nómbralo cómo `peticionGET2.txt`. Ahora edita la copia

realizada con el Notepad++ y añade las líneas 3 a 6, de forma que su contenido sea el que se muestra a continuación:

```
1 GET /sew/practicas/practica-01/ejemplo.html HTTP/1.1
2 Host: www-sew.dyndns.org
3 Accept-Encoding: gzip
4
5 GET /sew/practicas/practica-01/w3c.png HTTP/1.1
6 Host: www-sew.dyndns.org
7
```

Este contenido corresponde a dos peticiones GET consecutivas que solicitan los recursos identificados mediante las dos URLs que se indican a continuación:

http://www-sew.dyndns.org/sew/practicas/practica-01/ejemplo.html
http://www-sew.dyndns.org/sew/practicas/practica-01/w3c.png

Guarda el archivo, pero recuerda no dejar la marca EOF en la línea 7 porque en caso contrario la segunda petición quedaría incompleta y no se obtendría respuesta del servidor, y ahora utiliza el netcat como cliente para solicitar los recursos indicados.

PREGUNTAS

Si has preparado bien las peticiones (el archivo peticionGET2.txt) y has utilizado correctamente el comando nc en el intérprete de comandos, en la ventana de éste (salida estándar) se habrán volcado las respuestas de ambas peticiones. Estas respuestas constan de:

1. Una línea de estado que debería ser: HTTP/1.1 200 OK. Si alguna de las líneas de estado es distinta, probablemente la petición correspondiente sea errónea.
2. La cabecera del mensaje: lista de pares propiedad/valor separados por ‘:’.
3. Una línea en blanco.
4. El cuerpo del mensaje: texto correspondiente a la representación del recurso solicitado.

Si el cuerpo del mensaje es siempre texto (caracteres de 8 bits). ¿Cómo puede un cliente Web interpretar y, si fuera el caso, presentar de forma correcta el recurso solicitado?

Habrás observado que, en ambos casos, el cuerpo del mensaje de respuesta del servidor consta de caracteres ilegibles. ¿Cómo explicarías esto? En cada caso la respuesta debe ser precisa, por ejemplo, para el segundo caso la respuesta: *porque se está enviando una imagen al cliente*, no sería válida porque es demasiado imprecisa.

Discute las posibles respuestas a ambas preguntas con el compañero o compañeros que tengas al lado.

1.2.2 Netcat en modo servidor

Se va a utilizar ahora el netcat en modo servidor, para ello la orden nc debe incluir el parámetro -L y el número de puerto (-p *<número_puerto>*) por el que se queda a la escucha. Como se está analizando el protocolo HTTP, la idea es que el netcat haga la función de servidor Web que escucha por el puerto indicado. En realidad, el netcat carece de la mayor parte de las funcionalidades necesarias para ello, así que más bien

será un *pseudo-servidor Web*. En cuanto al número de puerto, se puede elegir uno cualquiera que no esté ocupado y superior a `1024` (los inferiores están reservados), por ejemplo, el puerto `12380`.

Tendremos entonces un *pseudo-servidor Web* escuchando por el puerto `12380` al que se podrán conectar los clientes Web, por ejemplo, un navegador. Lo habitual es que cliente y servidor residan en máquinas distintas y de hecho así podría ser, sin embargo, por simplicidad en esta prueba ambos residirán en la misma máquina: el PC local del usuario (ver Figura 7).

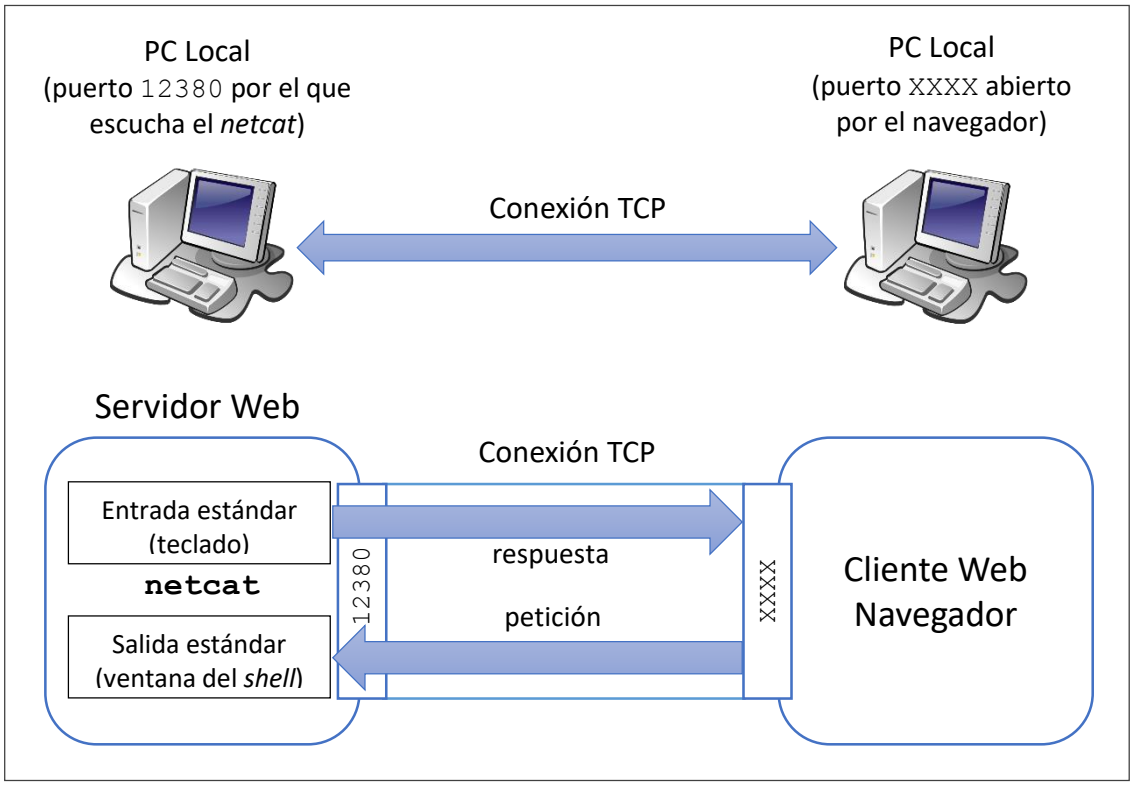

Figura 7. Conexión de un cliente al pseudo-servidor Web netcat

En este modo de funcionamiento, lo que puede recibir el `netcat` por el puerto a la escucha (`12380`) son las peticiones que los clientes Web (por ejemplo, un navegador) hagan al *pseudo-servidor Web* y esto será lo que se volcará a la salida estándar (la ventana del *shell*). Y para que el *pseudo-servidor Web* responda al cliente se debe proporcionar la respuesta escribiendo ésta en el puerto `12380` por medio de la entrada estándar (el teclado). Esto es lo que se ha reflejado de forma esquemática en la Figura 7.

Al igual que en el apartado 1.2.1, se redirigirá la entrada estándar a un archivo, si cabe aun con más motivo, ya que en general el formato del mensaje de respuesta HTTP es más complejo, o cuando menos más extenso, que el formato de una petición HTTP. El coste de esta redirección es que el *pseudo-servidor Web* emitirá una única respuesta, ya que lo que se escribe al puerto `12380` se lee de un archivo y éste sólo se puede leer una vez.

Como el *pseudo-servidor Web* residirá en el PC local del usuario, el puerto por el que estará a la escucha será el `12380` y se redirigirá la entrada estándar a un archivo, la forma más simple de que el navegador solicite un recurso es utilizar la URL:

http://localhost:12380/ o bien http://127.0.0.1:12380/

Observa que añadir una ruta cualquiera a la URL no tendrá efecto distinto alguno y que, en cualquier caso, la respuesta del *pseudo-servidor Web* siempre será la misma (el contenido del archivo redirigido).

Así pues, lo primero que se necesita es escribir en un archivo, que se nombrará como `respuesta.txt`, la respuesta HTTP del *pseudo-servidor Web*. Por otra parte, dado que a la salida estándar se vuelcan las peticiones del navegador, se aprovechará la ocasión para mostrar una petición `POST` completa y, para ello, el cuerpo del mensaje de respuesta será un documento HTML con un formulario interactivo. Concretamente el mismo documento que se obtiene al solicitar el recurso identificado por la URL:

http://www-sew.dyndns.org/sew/practicas/practica-01/ejemplo.php

y que ya se vio en el apartado 1.1.2.

Escribir toda la respuesta es tedioso y requiere conocimientos que, a priori, no se tienen, como es escribir un documento HTML con un formulario. Sin embargo, hay una forma simple de obtener ésta, que será la que utilizaremos: *con el* `netcat` *como cliente solicitar el recurso identificado por la URL indicada y redirigir la salida al archivo* `respuesta.txt`. Para preparar la petición del recurso, edita con el `Notepad++` el archivo `peticionGET.txt` que se encuentra en el directorio `practica-01` y cambia `ejemplo.html` por `ejemplo.php`, de forma que el contenido del archivo sea:

```
1 GET /sew/practicas/practica-01/ejemplo.php HTTP/1.1
2 Host: www-sew.dyndns.org
3
```

guarda el archivo (recuerda que la marca `EOF` no debe ir en la tercera línea) y, supuesto que el directorio de trabajo es `practica-01`, en el intérprete de comandos ejecuta la orden siguiente:

```
   Orden netcat        Host       Puerto  Redir. entrada   Redir. salida
$ netcat-1.11\nc www-sew.dyndns.org 80 <peticionGET.txt >respuesta.txt
```

Ahora en el archivo `respuesta.txt` ya tendremos el mensaje de respuesta HTTP y sólo resta editarlo con el `Notepad++` para quitar las cabeceras no necesarias, dejando únicamente las dos imprescindibles (`Content-Type` y `Content-Length`). Al editarlo ten cuidado de no hacer modificación alguna en el cuerpo del mensaje para no alterar su tamaño (valor de la propiedad `Content-Length`).

Pon ahora el `netcat` como servidor escuchando por el puerto `12380` y redirige la entrada estándar al archivo `respuesta.txt`, para lo cual, supuesto que `practica-01` es el directorio de trabajo, debes dar la siguiente orden en el intérprete de comandos:

```
   Orden netcat Serv. Puerto  Redirección
$ netcat-1.11\nc -L -p 12380 <respuesta.txt
```

Si ahora te conectas con un navegador al *pseudo-servidor Web* como que ya se ha indicado previamente, en la ventana del *shell* se volcará la petición del navegador y éste último presentará el documento con el formulario. Para finalizar, cubre el formulario y envía los datos recogidos al servidor, en la ventana del intérprete de comandos se mostrará el mensaje de la petición `POST` realizada. Observa que los datos viajan en el cuerpo de la petición y que éstos están codificados.

EJERCICIO 3 (OPCIONAL)

Prepara una petición condicional del recurso identificado mediante la URL:

http://www-sew.dyndns.org/sew/practicas/practica-01/w3c.png

utilizando la cabecera `If-None-Match`. Y utiliza el `netcat` para solicitar el recurso al servidor Web. Se deben obtener las dos posibles respuestas (en peticiones distintas), con las líneas de estado: `HTTP/1.1 304 Not Modified` y `HTTP/1.1 200 OK`. En el primero de los casos el cuerpo del mensaje es vacío (el servidor no envía el recurso porque se supone que está disponible en la caché del cliente), mientras que en el segundo caso el cuerpo del mensaje contiene la representación del recurso solicitado.