Práctica 2

Introducción

2. **¿Cuál es la función de la capa de aplicación?**

   - Define el formato de los mensajes: Existen protocolos que trabajan de forma binaria, por ejemplo usando ASN y otros en forma textual ASCII como HTTP.
   - Define la semántica de cada uno de los mensajes.
   - Define como debe ser el dialogo (intercambio de mensajes). Que mensajes se deben intercambiar.
   - Ejemplo concreto: Protocolo HTTP y sus implementaciones mediante servidores WEB y browsers (navegadores).

3. **Si dos procesos deben comunicarse:**

   a. **¿Cómo podrían hacerlo si están en diferentes máquinas?**

      Dos procesos de sistemas terminales distintos se comunican mediante el intercambio de mensajes a través de una red de computadoras. Uno de los procesos actúa como emisor, creando y enviando mensajes a la red, mientras que el otro proceso actúa como receptor, recibiendo estos mensajes y, posiblemente, respondiendo con mensajes propios.

   b. **Y si están en la misma máquina, ¿qué alternativas existen?**

      Cuando los procesos se ejecutan en el mismo sistema terminal, pueden comunicarse entre sí mediante sistemas de comunicación interprocesos, aplicando reglas gobernadas por el sistema operativo del sistema terminal.

4. **Explique brevemente cómo es el modelo Cliente/Servidor. De un ejemplo de un sistema Cliente/Servidor en la "vida cotidiana" y un ejemplo de un sistema informático que siga el modelo Cliente/Servidor. ¿Conoce algún otro modelo de comunicación?**

   Existe un host siempre activo, denominado servidor, con una dirección fija y conocida llamada dirección IP. Este servidor da (valga la redundancia) servicio a las solicitudes de muchos otros hosts, que son los clientes. Para ello el servidor recibe y sirve las solicitudes de los clientes (estos mismos no se comunican directamente entre sí).

Debido a que el servidor tiene una dirección fija y conocida, y siempre está activo, un cliente siempre puede contactar con él enviando un paquete a su dirección IP.

A menudo, en una aplicación cliente-servidor un único host servidor es incapaz de responder a todas las solicitudes de sus clientes. Por esta razón, en las arquitecturas cliente-servidor suele utilizarse un centro de datos, que alberga un gran número de hosts, para crear un servidor virtual de gran capacidad.

Un ejemplo de la vida cotidiana puede ser el intercambio entre cliente-vendedor en una tienda. Un ejemplo de un sistema informático que siga este modelo es una aplicación web, en donde el cliente sería el navegador web y el servidor sería el servidor web

Otro modelo de comunicación es P2P (Peer-to-Peer) en donde los dispositivos participantes son iguales y pueden actuar tanto como clientes como servidores para compartir recursos directamente entre sí, sin una infraestructura centralizada. Un ejemplo de P2P es BitTorrent.

5. **Describa la funcionalidad de la entidad genérica “Agente de usuario” o “User agent”**

   Un User Agent es la aplicación o software que actúa en nombre del usuario en un sistema o red. Es un componente que representa al usuario en sus interacciones con servicios en línea, sitios web u otros sistemas.

   En el contexto web, se refiere principalmente al navegador que un usuario emplea para acceder a sitios web. El User Agent comunica al servidor detalles del navegador, su versión, y a veces información sobre el sistema operativo y dispositivo. Esto permite que los sitios adapten su contenido a las capacidades del navegador.

   En una solicitud HTTP del navegador, el User Agent es un encabezado que informa sobre el navegador y sistema operativo.

HTTP

6. **Observe el indice de la RFC2616, busque el apartado donde se describe el requerimiento y la respuesta. ¿Qué son y en qué se diferencian HTML y HTTP? ¿En qué entidad ubicaría a HTML?**

   HTTP es el Protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol), que es el protocolo de la capa de aplicación de la Web. HTTP se implementa mediante dos programas: un programa cliente y un programa servidor. Ambos programas, que se ejecutan en sistemas terminales diferentes, se comunican entre sí intercambiando mensajes HTTP. HTTP define la

estructura de estos mensajes y cómo el cliente y el servidor intercambian los mensajes.

HTML, por otro lado, es el lenguaje de marcado utilizado para crear la estructura y el contenido de las páginas web. La mayoría de las páginas web están constituidas por un archivo base HTML y varios objetos referenciados. Por ejemplo, si una página web contiene texto HTML y cinco imágenes JPEG, entonces la página web contiene seis objetos: el archivo base HTML y las cinco imágenes. El archivo base HTML hace referencia a los otros objetos contenidos en la página mediante los URL de los objetos. Cada URL tiene dos componentes: el nombre de host del servidor que alberga al objeto y el nombre de la ruta al objeto.

El navegador web (cliente) es el que interpreta y renderiza el código HTML para mostrar la página web al usuario y el servidor web es el encargado de enviar los archivos HTML. El contenido HTML se encuentra en la entidad de respuesta, y es lo que el servidor envía al cliente como parte de la respuesta a la solicitud HTTP.

El envió y recibimiento de este mismo se hace a través de mensajes respetando el protocolo HTTP.

7. **Utilizando la VM, abra una terminal e investigue sobre el comando curl. Analice para qué sirven los siguientes parámetros (-I, -H, -X, -s).**

   curl es una herramienta para transferir datos desde o hacia un servidor, utilizando uno de los protocolos admitidos (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET y TFTP). El comando está diseñado para trabajar sin interacción del usuario.

   -I, --head
   (HTTP FTP FILE) Obtiene solo los encabezados. Los servidores HTTP cuentan con el comando HEAD que utiliza para obtener nada más que el encabezado de un documento. Cuando se usa en un archivo FTP o FILE, curl muestra solo el tamaño del archivo y la hora de la última modificación.

   -H, --header <header/@file>
   Sirve para incluir un header adicional en la solicitud HTTP. Se puede especificar cualquier número de encabezados adicionales. Hay que tener en cuenta que si se agrega un encabezado personalizado que tiene el mismo nombre que uno de los internos que usaría curl, se usará el encabezado establecido externamente en lugar del interno.

   -X, --request <command>
   Especifica un método de solicitud personalizado que se utilizará al comunicarse con el servidor HTTP. CURL por defecto utiliza GET.

   -s, --silent
   Modo silencioso de CURL. No mostrará la barra de progreso ni los mensajes de error. Silencia Curl. Seguirá generando los datos que solicita,

potencialmente incluso al terminal/salida estándar, a menos que los redirija. Es útil si deseas que la salida de curl sea más limpia y solo quieras ver la respuesta del servidor

8. **Ejecute el comando curl sin ningún parámetro adicional y acceda a www.redes.unlp.edu.ar. Luego responda:**

   a. **¿Cuántos requerimientos realizó y qué recibió? Pruebe redirigiendo la salida(>) del comando curl a un archivo con extensión html y abrirlo con un navegador**

      Realicé un requerimiento (método GET) y recibí un HTML.

   b. **¿Cómo funcionan los atributos href de los tags link e img en html?**

      El atributo href en la etiqueta <link> se utiliza para enlazar recursos externos y el atributo src en la etiqueta <img> se utiliza para especificar la ubicación de la imagen que se mostrará en la página web.

      En ambos casos, el navegador realiza solicitudes HTTP/HTTPS para descargar los recursos externos especificados en las URLs de los atributos href o src. Luego, dependiendo del tipo de recurso, realiza acciones adicionales como aplicar estilos en el caso de las hojas de estilo o mostrar imágenes en el caso de las etiquetas <img>.

   c. **Para visualizar la página completa con imágenes como en un navegador, ¿alcanza con realizar un único requerimiento? ¿Cuántos requerimientos serían necesarios para obtener una página que tiene dos CSS, dos Javascript y tres imágenes? Diferencie como funcionaría un navegador respecto al comando curl ejecutado previamente.**

      No, no alcanza con un único requerimiento. Cada recurso, como un archivo CSS, un archivo JavaScript o una imagen, se solicita por separado al servidor.

      Se necesitarán:

      1 requerimiento para el archivo base HTML
      1 requerimiento para el favicon
      2 requerimientos para CSS
      2 requerimientos para Javascript
      3 requerimientos para las tres imágenes

      Si el HTML tiene algún atributo href también se realizarán los requerimientos correspondientes para cada atributo.

      curl solo realizara un solo requerimiento (el archivo base HTML), requiere solicitudes manuales.

9. **Ejecute a continuación los siguientes comandos**:

   ```
   curl -v -s www.redes.unlp.edu.ar > /dev/null
   curl -I -v -s www.redes.unlp.edu.ar
   ```

   -v muestra información del intercambio que se está haciendo, información detallada de la comunicación entre el cliente y el servidor. (> es lo que el cliente le envio al servidor, < lo que el servidor respondió)

   - **Observe la salida y luego repita la prueba, pero previamente inicie una nueva captura en wireshark. Utilice la opción Follow Stream. ¿Qué se transmitió en cada caso?**

   - **¿A que se debió esta diferencia entre lo que se transmitió y lo que se mostró en pantalla?**

```
curl -v -s www.redes.unlp.edu.ar > /dev/null
```

```
GET / HTTP/1.1
Host: www.redes.unlp.edu.ar
User-Agent: curl/7.74.0
Accept: */*

HTTP/1.1 200 OK
Date: Mon, 28 Aug 2023 16:09:12 GMT
Server: Apache/2.4.56 (Unix)
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "1322-5f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 4898
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>.::.Redes y Comunicaciones.::.Facultad de Inform&aacute;tica.::.UNLP.::.</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="">
<meta name="author" content="">

<!-- Le styles -->
<link href="./bootstrap/css/bootstrap.css" rel="stylesheet">
<link href="./css/style.css" rel="stylesheet">
<link href="./bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

<!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
```

```
<!--[if lt IE 9]>
<script src="./bootstrap/js/html5shiv.js"></script>
<![endif]-->
</head>

<body>
<div id="wrap">
<div class="navbar navbar-inverse navbar-fixed-top">
<div class="navbar-inner">
<div class="container">
<a class="brand" href="./index.html"><i class="icon-home icon-white"></i></a>
<a class="brand" href="https://catedras.info.unlp.edu.ar" target="_blank">Redes y Comunicaciones</a>
<a class="brand" href="http://www.info.unlp.edu.ar" target="_blank">Facultad de Inform&aacute;tica</a>
<a class="brand" href="http://www.unlp.edu.ar" target="_blank">UNLP</a>
</div>
</div>
</div>

<div class="container">

<!-- Main hero unit for a primary marketing message or call to action -->
<div class="hero-unit">
<h2>Bienvenidos a Redes y Comunicaciones!</h2>
<p>Este CD es parte de los enunciados pr&aacute;cticos de la materia Redes y Comunicaciones de la carrera de Licenciatura en Inform&aacute;tica de la UNLP para la cursada del presente a&ntilde;o y servir&aacute; como herramienta para la realizaci&oacute;n de los trabajos pr&aacute;cticos.</p>
</div>

<div class="row">
<div class="span12">
<h3>Acerca de la VM</h3>
<p>Esta m&aacute;quina virtual est&aacute; basada en Debian GNU/Linux y fue creada por la c&aacute;tedra de Redes y Comunicaciones de la carrera de Licenciatura en Inform&aacute;tica de la UNLP para incluir las herramientas y configuraciones que se utilizar&aacute;n a lo largo de la cursada.</p>
<p>Se ha configurado al usuario <em><strong>root</strong></em> y al usuario <em><strong>redes</strong></em> con la misma contrase&ntilde;a: <strong>redes</strong>.</p>
</div>
</div>
<div class="row">
<div class="span12">
<h3>Ejercicios Pr&aacute;cticos</h3>
<p>Todo el material se va a encontrar publicado en el sitio de la c&aacute;tedra en <a href="https://catedras.info.unlp.edu.ar/" target="_blank">https://catedras.info.unlp.edu.ar/</a>.</p>
</div>
```

```
</div>
<div class="row">
<div class="span2">
<h4>Introducci&oacute;n</h4>
<p>
<ul>
<li>Nociones b&aacute;sicas</li>
</ul>
</p>
</div>
<div class="span3">
<h4>Capa de Aplicaci&oacute;n</h4>
<p>
<ul>
<li><a href="http/protocolos.html">Protocolos HTTP</a></li>
<li><a href="http/metodos.html">M&eacute;todos HTTP</a></li>
</ul>
</p>
</div>
<div class="span3">
<h4>Capa de Transporte</h4>
<p>
<ul>
<li>TCP</li>
<li>UDP</li>
</ul>
</p>
</div>
<div class="span2">
<h4>Capa de Red</h4>
<p>
<ul>
<li>IP</li>
<li>
Algoritmos de ruteo:<br/>
Topolog&iacute;as CORE:
<ul>
<li><a href="./core/1-ruteo-estatico.xml" target="_blank">Est&aacute;tico</a></li>
<li><a href="./core/2-ruteo-RIP.xml" target="_blank">RIP</a></li>
<li><a href="./core/3-ruteo-OSPF.xml" target="_blank">OSPF</a></li>
</ul>
</li>
<li>ICMP</li>
</ul>
</p>
</div>
<div class="span2">
<h4>Capa de Enlace</h4>
<p>
<ul>
```

```
<li>ARP</li>
<li>
Switch - Hub
</li>
</ul>
</p>
</div>
</div>

<div class="row">
<hr>
<p>Desarrollado originalmente por Christian Rodriguez y Paula Venosa en el a..o 2007, modificado en 2013 por el grupo de desarrollo de Lihuen GNU/Linux y en 2016 y 2020 por Leandro Di Tommaso.</p>
<br/>
</div>
</div>
</div>
<div id="footer">
<div class="container">
<p class="muted credit">Redes y Comunicaciones</p>
</div>
</div>


</body>
</html>
```

curl -I -v -s www.redes.unlp.edu.ar

```
HEAD / HTTP/1.1
Host: www.redes.unlp.edu.ar
User-Agent: curl/7.74.0
Accept: */*

HTTP/1.1 200 OK
Date: Mon, 28 Aug 2023 16:12:19 GMT
Server: Apache/2.4.56 (Unix)
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "1322-5f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 4898
Content-Type: text/html
```

La diferencia se debe a > /dev/null que redirige la salida estandar(la respuesta enviada por el servidor) a /dev/null (que la descarta)

10. Investigue cómo define las cabeceras la RFC.

a. Establece todas las cabeceras posibles?.

La RFC no establece todas las cabeceras posibles ya que se pueden utilizar cabeceras personalizadas, pero proporciona una lista de cabeceras estándar y comunes que se utilizan en el protocolo HTTP. Algunas de estas cabeceras incluyen "Date", "Content-Type", "Content-Length", "User-Agent", "Host", "Accept", "Authorization" y muchas más. Los desarrolladores tienen la flexibilidad de definir cabeceras personalizadas según las necesidades de sus aplicaciones, pero se espera que sigan las convenciones y estándares definidos en la RFC.

b. ¿Cuántas cabeceras viajaron en el requerimiento y en la respuesta del ejercicio anterior?

3 en el requerimiento
7 en la respuesta

c. ¿La cabecera Date es una de las definidas en la RFC? ¿Qué indica?

Sí, la cabecera "Date" es una de las cabeceras definidas en la RFC. Esta cabecera indica la fecha y hora en que se generó la solicitud o respuesta HTTP.

11. Utilizando curl, realice un requerimiento con el método HEAD al sitio www.redes.unlp.edu.ar e indique:

a. **¿Qué información brinda la primer línea de la respuesta?**

La primera línea brinda la versión de HTTP (HTTP/1.1), el código de estado (200) y la reason phrase (OK)

b. ¿Cuántos encabezados muestra la respuesta?

7

c. ¿Qué servidor web está sirviendo la página?

Apache/2.4.56 (Unix)

d. ¿El acceso a la página solicitada fue exitoso o no?

Si, fue exitoso ya que el código de estado es 200 OK, que es el código de respuesta de estado satisfactorio

e. **¿Cuándo fue la última vez que se modificó la página?**

El 19 de marzo de 2023 a las 16:04:46 (19:04:46 GMT)

f. **Solicite la página nuevamente con curl usando GET, pero esta vez indique que quiere obtenerla sólo si la misma fue modificada en una fecha posterior a la que efectivamente fue modificada. ¿Cómo lo hace? ¿Qué resultado obtuvo? ¿Puede explicar para qué sirve?**

Lo hago haciendo uso del header “If-Modified-Since”:<fecha>

```
curl -H "If-Modified-Since:Sun, 19 Mar 2023 19:04:46 GMT"
www.redes.unlp.edu.ar
```

Si se quiere acceder a una página recién cuando sea modifica y esta pesara (mucho), nos ahorramos todo el intercambio en el caso de que no haya sido modificada, ya que es la misma que tendríamos guardada.

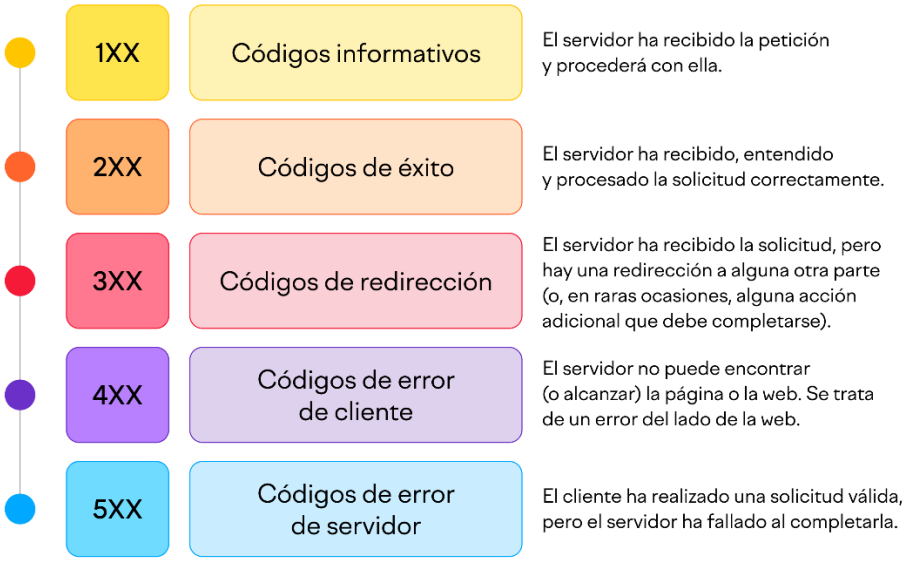
12. **En HTTP/1.0, ¿cómo sabe el cliente que ya recibió todo el objeto solicitado completamente? ¿Y en HTTP/1.1?**

En HTTP/1.0:
El cliente se da cuenta de que ha recibido todo el objeto solicitado cuando el servidor cierra la conexión después de enviar la respuesta.

En HTTP/1.1:
Se introducen mejoras para manejar este problema. El encabezado "Content-Length" indica al cliente la longitud en bytes del objeto en la respuesta, permitiéndole saber cuántos datos esperar. El encabezado "Transfer-Encoding" con valor "chunked" divide la respuesta en trozos, y el cliente detecta el final de la respuesta cuando recibe un trozo de tamaño 0.

13. **Investigue los distintos tipos de códigos de retorno de un servidor web y su significado en la RFC. ¿Qué parte se ve principalmente interesada de esta información, cliente o servidor? ¿Es útil que esté detallado y clasificado en la RFC?. Dentro de la VM, ejecute los siguientes comandos y evalue el estado que recibe**

- **curl -I http://unlp.edu.ar**
- **curl -I www.redes.unlp.edu.ar/restringido/index.php**
- **curl -I www.redes.unlp.edu.ar/noexiste**



Ejemplos:

- 200 OK
- 301 Moved Permanently
- 400 Bad Request
- 403 Access Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 500 Internal Server Error (CGI Error)
- 501 Method Not Implemented

Tanto los clientes como los servidores se benefician al entender los códigos de respuesta en el proceso de comunicación web. El cliente puede saber si su solicitud fue procesada con éxito o si hubo algún error, mientras que el servidor utiliza el código de retorno para comunicar el resultado de la solicitud al cliente y de alguna manera orientarlo sobre cómo interpretar la respuesta.

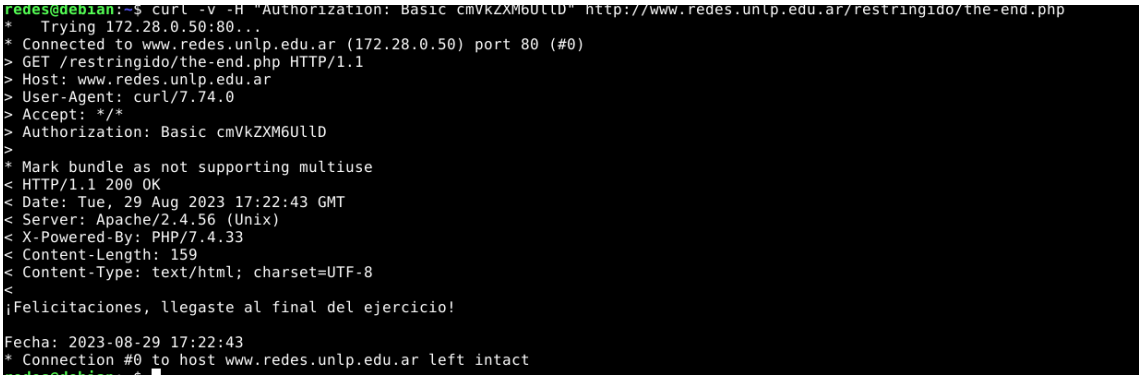
Si, es útil que este detallado ya que establece un estándar común para la comunicación entre clientes y servidores en la web asegurando respuestas coherentes e interoperables de acuerdo a la situación.

```
redes@debian:~$ curl -I http://unlp.edu.ar
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0
Date: Tue, 29 Aug 2023 17:13:05 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://unlp.edu.ar/

redes@debian:~$ curl -I www.redes.unlp.edu.ar/restringido/index.php
HTTP/1.1 401 Unauthorized
Date: Tue, 29 Aug 2023 17:13:23 GMT
Server: Apache/2.4.56 (Unix)
WWW-Authenticate: Basic realm="Authentication Required"
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "cb-5f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 203
Content-Type: text/html

redes@debian:~$ curl -I www.redes.unlp.edu.ar/noexiste
HTTP/1.1 404 Not Found
Date: Tue, 29 Aug 2023 17:13:35 GMT
Server: Apache/2.4.56 (Unix)
Content-Type: text/html; charset=iso-8859-1
```

14. **Utilizando curl, acceda al sitio www.redes.unlp.edu.ar/restringido/index.php y siga las instrucciones y las pistas que vaya recibiendo hasta obtener la respuesta final. Será de utilidad para resolver este ejercicio poder analizar tanto el contenido de cada página como los encabezados**

```
redes@debian:~$ curl http://www.redes.unlp.edu.ar/restringido/index.php
<h1>Acceso restringido</h1>

<p>Para acceder al contenido es necesario autenticarse. Para obtener los datos de acceso seguir las instrucciones detalladas en www.red
es.unlp.edu.ar/obtener-usuario.php</p>
redes@debian:~$ curl www.redes.unlp.edu.ar/obtener-usuario.php
<p>Para obtener el usuario y la contraseña haga un requerimiento a esta página seteando el encabezado 'Usuario-Redes' con el valor 'obt
ener'</p>
redes@debian:~$ curl -H "Usuario-Redes:obtener" www.redes.unlp.edu.ar/obtener-usuario.php
<p>Bien hecho! Los datos para ingresar son:

    Usuario: redes

    Contraseña: RYC

    Ahora vuelva a acceder a la página inicial con los datos anteriores.

    PISTA: Investigue el uso del encabezado Authorization para el método Basic. El comando base64 puede ser de ayuda!</p>
redes@debian:~$ echo -n "redes:RYC" | base64
cmVkZXM6UllD
```

```
redes@debian:~$ curl -v -H "Authorization: Basic cmVkZXM6UllD" http://www.redes.unlp.edu.ar/restringido/index.php
*   Trying 172.28.0.50:80...
* Connected to www.redes.unlp.edu.ar (172.28.0.50) port 80 (#0)
> GET /restringido/index.php HTTP/1.1
> Host: www.redes.unlp.edu.ar
> User-Agent: curl/7.74.0
> Accept: */*
> Authorization: Basic cmVkZXM6UllD
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 302 Found
< Date: Tue, 29 Aug 2023 17:22:15 GMT
< Server: Apache/2.4.56 (Unix)
< X-Powered-By: PHP/7.4.33
< Location: http://www.redes.unlp.edu.ar/restringido/the-end.php
< Content-Length: 230
< Content-Type: text/html; charset=UTF-8
<
<h1>Excelente!</h1>

<p>Para terminar el ejercicio deberás agregar en la entrega los datos que se muestran en la siguiente página.</p>
<p>ACLARACION: la URL de la siguiente página está contenida en esta misma respuesta.</p>

* Connection #0 to host www.redes.unlp.edu.ar left intact
```



***NOTAS:**

El método de autenticación "**Basic**" en HTTP es una forma simple de autenticar a los usuarios utilizando credenciales (nombre de usuario y contraseña). Se utiliza en combinación con HTTPS para mayor seguridad.
El motivo por el cual las credenciales se codifican en base64 es principalmente para evitar problemas con caracteres especiales que podrían afectar la comunicación HTTP. Base64 es una forma de representar datos binarios en una cadena de caracteres ASCII, lo que lo hace seguro para transmitir a través de HTTP.

El encabezado "**Location**" en HTTP se utiliza para redirigir al cliente a una ubicación diferente. Cuando un servidor envía una respuesta con un encabezado "Location", está indicando al cliente que realice una nueva solicitud a la URL especificada en ese encabezado.

15. Utilizando la VM, realice las siguientes pruebas:

a. **Ejecute el comando ’curl www.redes.unlp.edu.ar/extras/prueba-http-1-0.txt’ y copie la salida completa (incluyendo los dos saltos de linea del final).**
b. **Desde la consola ejecute el comando telnet www.redes.unlp.edu.ar 80 y luego pegue el contenido que tiene almacenado en el portapapeles. ¿Qué ocurre luego de hacerlo?**

```
redes@debian:~$ telnet www.redes.unlp.edu.ar 80
Trying 172.28.0.50...
Connected to www.redes.unlp.edu.ar.
Escape character is '^]'.
GET /http/HTTP-1.1/ HTTP/1.0
User-Agent: curl/7.38.0
Host: www.redes.unlp.edu.ar
Accept: */*



HTTP/1.1 200 OK
Date: Tue, 29 Aug 2023 17:38:03 GMT
Server: Apache/2.4.56 (Unix)
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "760-5f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 1888
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Protocolo HTTP: versiones</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">

    <!-- Le styles -->
    <link href="../../bootstrap/css/bootstrap.css" rel="stylesheet">
    <link href="../../css/style.css" rel="stylesheet">
    <link href="../../bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

    <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
      <script src="./bootstrap/js/html5shiv.js"></script>
    <![endif]-->
  </head>

  <body>


    <div id="wrap">

    <div class="navbar navbar-inverse navbar-fixed-top">
      <div class="navbar-inner">
        <div class="container">
          <a class="brand" href="../../index.html"><i class="icon-home icon-white"></i></a>
          <a class="brand" href="https://catedras.info.unlp.edu.ar" target="_blank">Redes y Comunicaciones</a>
          <a class="brand" href="http://www.info.unlp.edu.ar" target="_blank">Facultad de Inform&aacute;tica</a>
          <a class="brand" href="http://www.unlp.edu.ar" target="_blank">UNLP</a>
        </div>
      </div>
    </div>

    <div class="container">
    <h1>Ejemplo del protocolo HTTP 1.1</h1>
    <p>
        Esta p&aacute;gina se visualiza utilizando HTTP 1.1. Utilizando el capturador de paquetes analice cuantos flujos utiliza el navegador para visualizar la p&aacute;gina con sus im&aacute;genes en contraposici&oacute;n con el protocolo HTTP/1.0.
    </p>
    </p>
    <h2>Imagen de ejemplo</h2>
    <img src="13532-tuxkiller03green.png" width="800px"/>
    </div>
```

c. Repita el proceso anterior, pero copiando la salida del recurso /extras/prueba-http-1-1.txt. Verifique que debería poder pegar varias veces el mismo contenido sin tener que ejecutar telnet nuevamente.

El resultado es el mismo, la diferencia que hay es que la conexión no se cierra una vez que el servidor me envia la respuesta.

16. En base a lo obtenido en el ejercicio anterior, responda:

- **¿Qué está haciendo al ejecutar el comando telnet?**

  El comando intenta establecer una conexión a través del protocolo Telnet con el servidor en "www.redes.unlp.edu.ar" en el puerto 80.

- **¿Qué lo diferencia con curl? Observe la definición de método y recurso en la RFC. Luego responda, ¿Qué método HTTP utilizó?**

  Telnet se centra en la comunicación interactiva con servidores a través del protocolo Telnet, curl se centra en la transferencia de datos utilizando diversos protocolos, incluidos los seguros.
  Se utilizo el método GET
  GET /http/HTTP-1.1/ HTTP/1.0
  User-Agent: curl/7.38.0
  Host: www.redes.unlp.edu.ar
  Accept: */*

- **¿Qué recurso solicitó? ¿Qué diferencias notó entre los dos casos? ¿Puede explicar por qué?**

Solicito el recurso /http/HTTP-1.1/. La diferencia que hay es que la conexión no se cierra una vez que el servidor envia la respuesta en el punto c. Esto se debe a que en contraste con HTTP 1.0, donde cada solicitud y respuesta requería abrir y cerrar una nueva conexión, en HTTP 1.1, una sola conexión puede ser reutilizada para varias transacciones.

- **¿Cuál de los dos casos le parece más eficiente? Piense en el ejercicio donde analizó la cantidad de requerimientos necesarios para obtener una página con estilos, javascripts e imágenes. El caso elegido, ¿puede traer asociado algún problema?**

  Es más eficiente el caso de HTTP 1.1 ya que reduce la sobrecarga asociada con la apertura y el cierre de conexiones, lo que conduce a una carga más rápida de páginas web y una mejor utilización de los recursos de red.

  Hay un problema asociado con la conexión persistente, especialmente cuando se trata de recursos grandes o de larga duración. Si un recurso se mantiene en la conexión durante un período prolongado, podría bloquear la conexión y retrasar la obtención de otros recursos. Esto se conoce como el problema de "holgura de cabeza" (head-of-line blocking). Para abordar este problema, los navegadores modernos utilizan técnicas como la multiplexación y el pipelining para permitir la transferencia de múltiples recursos a la vez.