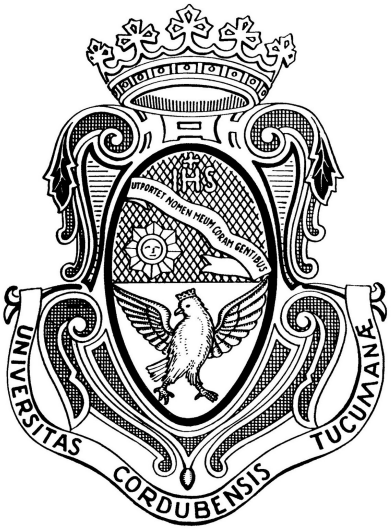
UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS FÍSICAS Y NATURALES



SISTEMAS OPERATIVOS II

TRABAJO PRÁCTICO N°3: Sistemas embebidos - Servidor web

Integrantes:

- Cazajous Miguel A. - 34980294

Córdoba - Argentina

2 de febrero de 2019

Índice

1. Introducción:

1.1. Propósito:

El propósito es el desarrollo de una aplicación que permita obtener información acerca del sistema donde se corre un servidor, entre otros datos relevantes.

1.2. Ambito del sistema:

La aplicación desarrollada está pensada para correr en un servidor bajo un Sistema Operativo GNU/Linux (el más común en sistemas embebidos).
Del lado del cliente la interfaz para usar la aplicación la provee un navegador web cualquiera, de manera que cualquier computadora que sea capaz de conectarse en red y disponga de un navegador debería ser capaz de obtener información y realizar operaciones a distancia en el servidor. La página web que provee el servidor dispondrá, para el usuario que ingresa mediante el navegador, una interfaz que le permitirá al cliente obtener información del mismo.

1.3. Definiciones, acrónimos y abreviaturas:

1. GNU/Linux: Sistema Operativo que tiene sus orígenes en Unix y debe su nombre al uso del kernel de Linux y el uso de herramientas GNU.
2. Git: Sistema de control de versiones. Lleva un registro de todos los cambios efectuados en los archivos y permite volver a versiones anteriores fácilmente.
3. Software: Referente a programas, aplicaciones.
4. Hardware: Componentes físicos, computadora, placa de desarrollo, etc.
5. Sistema Operativo: Software principal residente en un hardware que permite la gestión de sus partes y su interacción con aplicaciones de usuario. Es una capa intermedia entre aplicaciones de usuario y el hardware de la máquina.
6. Ethernet: Estándar de transmisión de datos.
7. SSH: Protocolo de conexión de máquinas remotas a través de una red.
8. Kernel: Núcleo, sistema de software base, encargado de la gestión de los recursos de hardware.

9. Módulo: Agregado de un kernel que provee nuevas funcionalidades.
10. Página web: Información (texto, audio, video, imágenes) que pueden ser obtenidos mediante un navegador web.
11. Navegador web: Software encargado del acceso a la red que permite la lectura de diferentes páginas web.
12. Servidor web: Véase apéndices.

1.4. Referencias:

[1] w3schools. HTML. URL: https://www.w3schools.com.

[2] Lucid Software Inc. LucidChart. URL: https://www.lucidchart.com.

[3] Tutorials point. Perl and Cgi tutorial. URL: https://www.tutorialspoint.com/perl/perl_cgi.htm.

[4] Jon Allen. CGI. URL: http://perldoc.perl.org/CGI.html.

[5] Jon Allen. Perlre. URL: https://perldoc.perl.org/perlre.html.

[6] tlpd.org. Compiling Kernel Modules. URL: http://www.tldp.org/LDP/lkmpg/2.6/html/x181.html.

[7] The Linux Foundation. Light Weight, open source web servers. URL: https://www.linux.com/news/which-light-weight-open-source-web-server-right-you.

[8] Wikipedia. Servidor Web. URL: https://es.wikipedia.org/wiki/Servidor_web.

[9] Net Reliant. Compacting VirtualBox Disk Images - Linux Guests. URL: http://www.netreliant.com/news/8/17/Compacting-VirtualBox-Disk-Images-Linux-Guests.html.

[10] Wikipedia. Comparison of web server software. URL: https://en.wikipedia.org/wiki/Comparison_of_web_server_software.

[11] Net Reliant. Compacting VirtualBox Disk Images - Linux Guests. URL: https://tools.ietf.org/html/rfc1867.

1.5. Descripción general del documento:

Este documento se compone de 7 secciones principales, siendo la primera la introducción, donde se explica brevemente el fin del proyecto como así también sus requerimientos de forma muy general. En la segunda sección la

descripción general del sistema con el fin de conocer las principales funciones que debe ser capaz de realizar, conocer a quién va dirigido, además de restricciones y supuestos que puedan afectar el desarrollo del mismo. En la sección 3 se definen de manera detallada los requerimientos del sistema a desarrollar, los que definen el comportamiento del sistema como así también otros requerimientos que puedan ser deseados considerando el uso que el sistema va a tener. En la cuarta sección se presenta el diseño del sistema que dará solución a los requerimientos antes enumerados. En la quinta sección se discuten los resultados de implementación del diseño elaborado en la etapa anterior donde se muestra como el sistema opera. En la sección 6 se elabora una breve conclusión acerca de la experiencia en la elaboración del proyecto. Finalmente en la última sección se agrega información alternativa que pueda ser de interés.

2. Descripción general:

2.1. Perspectiva del producto:

La aplicación deberá proveer un método rápido y sencillo de acceso a la información de manera remota mediante una página web con el fin de mejorar la disponibilidad de información.

2.2. Funciones del producto:

El sistema deberá ser capaz de:

1. Desplegar un menú de opciones mediante botones o cuadros de diálogo que le permita al cliente operar con el servidor.

2. El servidor debe ser capaz de interpretar las peticiones y tomar decisiones adecuadas a estos. Además en ciertas operaciones se requiere que el servidor web tenga permisos de privilegio para ejecutar tareas en el sistema embebido.

2.3. Características de los usuarios:

La aplicación está destinada a usuarios autorizados a obtener información del servidor, aunque en este trabajo no hay mayores medidas de seguridad que vayan a ser incluidas, para verificar que quien quiera ingresar a obtener información, sea o no, alguien con autorización. Cualquier operador que esté familiarizado con una página web es apto para operar.

2.4. Restricciones:

Como restricción de lenguaje de programación se tiene que la aplicación debe ser en su totalidad elaborada bajo lenguage C, Perl y/o HTML. Además como se mencionó el entorno donde corre la aplicación (servidor) debe ser cualquier distribución basada en GNU/Linux.

2.5. Suposiciones y dependencias:

Un cambio en el sistema operativo donde se desea que corra la aplicación (servidor web) requeriría de cambios sustanciales en la funcionalidad del producto, así también como en el desarrollo del mismo. El hardware como se mencionó tiene un gran rango de variación aceptable que no generaría ningún inconveniente en el uso de la aplicación, aunque quizás haya diferencias referidas a la compilación de módulos, dependiendo del sistema operativo que esté disponible para ese hardware.

2.6. Requisitos futuros:

El proyecto tiene un fin específico y no está pensado para abarcar más requerimientos en el futuro, de todas maneras el desarrollo de páginas web es muy flexible y para un caso que no requiera demasiado desarrollo una modificación sería muy simple. La mayor complejidad está en el código C y/o Perl que es donde se manipula la información. HTML es más que todo para presentación.

3. Requisitos específicos:

3.1. Interfaces externas:

Interfaz de software:

La interfaz de software será gráfica, y la misma contará de cuadros de diálogos y botones que responderán a lo que el operador quiera solicitar y será accedida desde un navegador web.

Interfaz de hardware y comunicación:

La interfaz de hardware y comunicación consta de una computadora (cliente) que disponga de un navegador (independientemente de que Sistema Operativo corra) y una plataforma de desarrollo bajo GNU/Linux conectada a la computadora mediante una interfaz Ethernet.

Figura 1: Conexión a modo de ilustración de una placa de desarrollo Intel Galileo y una notebook

3.2. Funciones:

El sistema deberá ser capaz de:

1. Mostrar una interfaz con las siguientes opciones:
   a) Información acerca de memoria
   b) Información acerca de procesador
   c) Uptime
   d) Fecha y hora del servidor
   e) Ingresar comandos: Cuadro de diálogo que permite enviar comandos al bash del servidor.
   f) Listar módulos del servidor
   g) Formulario para cargar un archivo .ko
   h) Instalación de un módulo
   i) Desinstalación de un módulo

3.3. Requisitos de rendimiento:

La aplicación es interactiva y se espera que el tiempo de respuesta sea el mínimo posible para cualquier petición que se realice, siendo deseable que se obtengan resultados en milésimas de segundo.

3.4. Restricciones de diseño:

Las únicas restricciones de diseño que se tienen al momento de elaborar este documento es la restricción de los lenguage de programación que se deben usar.

3.5. Atributos del sistema:

El sistema de seguridad en este caso está practicamente ausente, aunque no es lo aconsejable que así sea.

El sistema es fácilmente portable a cualquier computadora y plataforma de desarrollo que cumpla con los requisitos.

La mantenibilidad del softwre es sencilla ya que el mismo lleva consigo un control de versiones implementado mediante la herramienta Git.

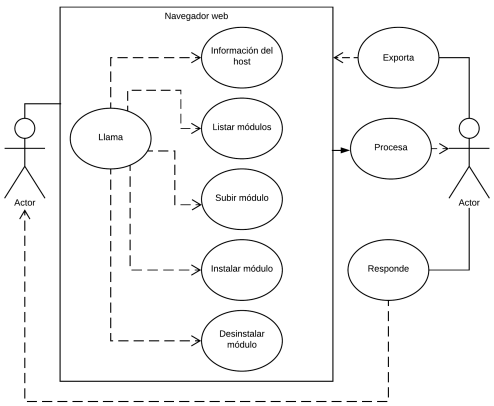
Factores que pueden afectar principalmente la fiabilidad del software están en un fallo del SO tanto en la computadora, pero mayormente en la plataforma de desarrollo que correrá el sistema servidor y también en un fallo entre la conexión de las anteriores mencionadas.

4. Otros requisitos:

No se tienen otros requisitos presentes que no hayan sido ya mencionados en las secciones anteriores.

5. Diseño de solución:

Representación de los requerimientos mediante: casos de uso: (Click en la imagen para abrir)



6. Implementación y resultados:

6.1. Paso a paso la implementación del servidor web y de la página corriendo:

NOTA: Todo el desarrollo que se mostrará a continuación fue hecho en una Raspberry PI corriendo el sistema operativo Raspbian.

Acceso

1. Usuario: pi
2. Contraseña: raspberry

1. Instalación:

   Para la instalación del servidor web "lighttpd" se tipeó simplemente.

   ```
   sudo apt install lighttpd
   ```

2. Configuración:

   Dentro de la carpeta */etc/lighttpd* existe un archivo de configuración que debe modificarse para que el servidor sea capaz de ejecutar scripts CGI.

   El archivo en cuestión es lighttpd.conf.

   En el mismo se deben agregar las siguientes líneas.
   Agregar o descomentar.

   ```
   “mod_cgi”
   ```

   Luego agregar lo siguiente

   ```
   $HTTP[”url”] =~ “^”{
     cgi.assign = (“.pl” => “/usr/bin/perl”)
   }
   ```

   Donde “^” representa la carpeta donde residen los scripts (también configurable) que por defecto es */var/www/http*

3. Compilación de módulos:

   De no ser posible compilar el módulo en el hardware del servidor, debe usarse compilación cruzada en otro hardware.

   Se implementó un módulo simple que solo muestra “hello world” en el kernel cuando se instala y “goodbye world” cuando se remueve.

   La compilación se hace mediante un Makefile que acompaña al código en C, dando como resultado un archivo .ko que es el que nos interesa (*La compilación debe hacerse en un ruta sin espacios, de lo contrario dará error.*).

   Para la compilación del módulo es necesario instalar los headers del kernel. Para eso se procede como sigue.

   ```
   sudo apt-get install raspberrypi-kernel-headers
   ```

   Los pasos para instalar y remover el módulo son:

   ```
   sudo insmod hello_world.ko
   sudo rmmod hello_world
   ```

   Puede ser necesario utilizar la ruta absoluta de los comandos anteriores

Mediante el comando.

```
dmesg
```

Podemos ver los mensajes que el módulo escribió en el Kernel.

4. Módulos de Perl:

   Fue necesario instalar el módulo CGI de Perl. La instalación en Raspbian mediante CPAN es:

   ```
   sudo cpan install CGI
   ```

5. Resultados:

   Nos conectamos mediante un navegador web usando la IP del servidor.

Figura 2: Navegador “opera” conectándose al servidor.

Información del hardware del servidor, uptime y fecha.

Figura 3: Memoria

Figura 4: Procesador

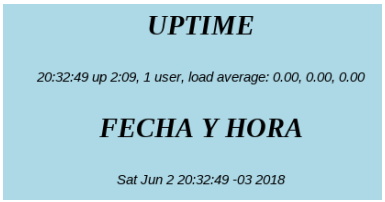


Figura 5: Uptime y fecha

Vemos ahora algunos ejemplos de salida de comandos enviados al bash del servidor

Figura 6: Comando “ls -ls”

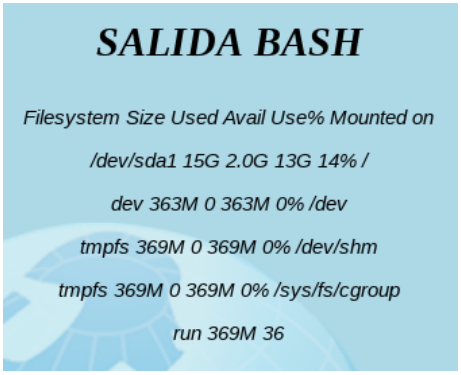


Figura 7: Comando “df -h”

Figura 8: Comando “pwd”

Subir un módulo:

Figura 9: Ventana que permite elegir un archivo.

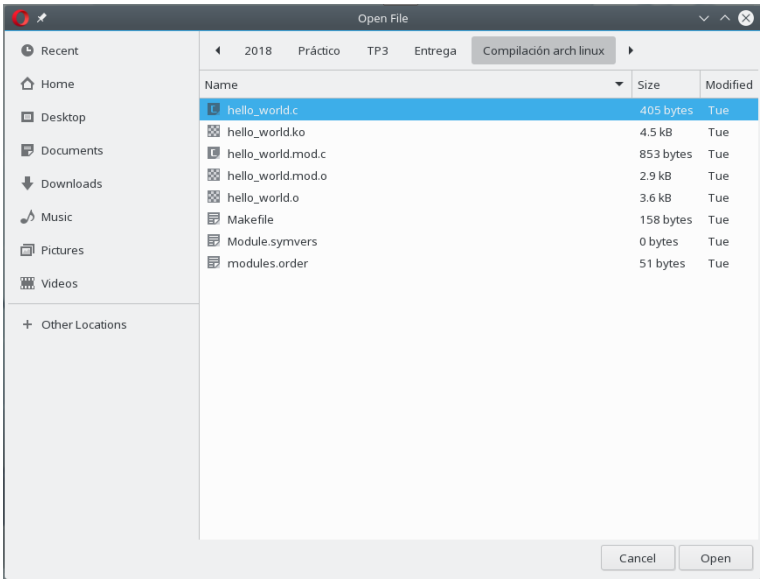


Figura 10: Se muestra aquí un explorador de archivos que permite elegir uno del disco.

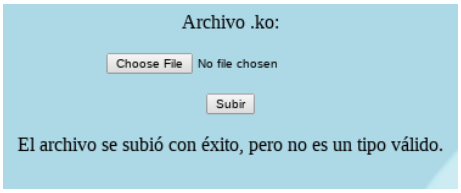


Figura 11: Sube el archivo, pero muestra un error si no es del tipo .ko.

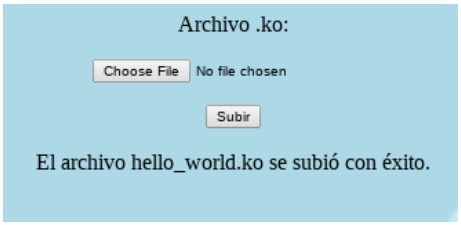


Figura 12: Sube el archivo y reconoce el formato como válido.

Instalación del módulo subido

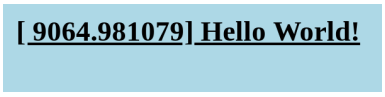


Figura 13: Muestra el mensaje que se imprime en el kernel.

Muestra los módulos instalados

Figura 14: Se ve en la primer línea el módulo que se acaba de instalar.

Desinstalación del módulo

```
[ 9103.392131] Goodbye World!
```

Figura 15: Muestra el mensaje que se imprime en el kernel.

6. Imagen:

   Para armar la imagen se utilizó el siguiente comando, de manera que la imagen no fuera exageradamente grande, ya que consideraba el espacio libre de la tarjeta SD.

```
sudo dd if=/dev/* | gzip > imagen.img.gz
```

   Enlace a la imagen en Mega.nz

   Imagen Raspbian

7. Conclusiones:

Se logró con este trabajo un entendimiento general del funcionamiento de un servidor web y que es lo que ocurre por detrás cuando se navega por internet mediante un navegador web, aunque sea a un nivel muy general.
Se adquirieron nuevos conocimientos en el manejo de sistemas embebidos por el simple hecho de la resolución de problemas que van surgiendo a medida se elaboraba el trabajo y que no estaban especificados en los requerimientos.
Se consiguió una muy buena interacción entre lenguajes C, Perl y HTML funcionando juntos y de este último se aprendió como puede dársele a una página web una interfaz mucho más amigable o estéticamente más agradable mediante la incorporación de CSS sin necesidad de saber mucho en la materia de programación web.

8. Apéndices:

8.1. Servidores web:

Un servidor web es un software que procesa un programa del lado del servidor mediante conexiones con un cliente, que genera una respuesta que es interpretada por el navegador del cliente.

La consulta del cliente al servidor se denomina petición web y como se dijo la misma se realiza a través de un navegador web.

Dentro del código (más común HTML) para la petición se puede elegir entre dos formularios, GET y POST.

```
<form action="/action_page.php" method="">
First name: <input type="text" name="fname"><br>
Last name: <input type="text" name="lname"><br>
<input type="submit" value="Submit">
</form>
```

Donde en method="" se reemplaza la cadena vacía por GET o POST.
La mayor diferencia está en el hecho que POST oculta la información de envío, esto puede verse claramente con un simple ejemplo donde se selecciona ambos métodos, observando que en el caso de GET al lado de la página (ej: 192.168.1.1/funcion.pl) se encuentra información aparte.

El procesamiento de ambos formularios del lado del servidor también es diferente, siendo en el caso de GET más simple.

El método GET se usó para todas las peticiones que se realizaron en este trabajo.

Puede decirse también que entre los métodos GET y POST como sus respectivos nombres lo indican, el primero es más para adquirir información y el segundo para enviar información.

8.2. Elección de un servidor web:

Existen innumerables servidores web de los que se pueda hacer uso. Para nuestro caso, teniendo en cuenta que el mismo se iba a implementar en una placa de desarrollo, se buscó que el servidor no sea demasiado complejo, ya que demandaría más espacio para su uso y así también más consumo puesto que estos proveen de muchas funcionalidades que para nosotros no son de

interés.

De modo que la elección se enfoca más en escoger un servidor web liviano que posea lo mínimo e indispensable.

Entendiéndose como que sea capaz de procesar CGI (Common Gateway Interface) en cual es un método para el intercambio de información que permite una interacción más dinámica que mediante HTML.

Dentro de CGI nos interesaría que el servidor sea capaz de tratar con archivos .pl, es decir scripts realizado en el lenguaje Perl, el cual posee un módulo CGI.

Estos archivos están guardados en el servidor esperando una llamada de HTML que es del tipo.

```
<form action=''/script.pl'' methos=''GET''>
...
```

A continuación se muestra un cuadro que resume las características de solo algunos de los tantos servidores web disponibles para su uso.

Antes de comenzar con la elección del servidor web, se buscó cuales permitían CGI y cuales estaban disponibles para GNU/Linux [10]. Las opciones anteriores no son un buen filtro para elegir servidor web, pues la gran mayoría cuenta con esos dos aspectos, de manera que se incluyeron algunas restricciones más en la elección orientadas a la facilidad, disponibilidad y capacidades de hardware donde el servidor web se va a ejecutar.

Cuadro comparativo:

Servs. Web	Instalación	CGI	Configuración	Funciona?	Minimalismo
Nginx	Se encuentra en el repositorio oficial y se puede instalar mediante apt	No. Solo Fast CGI	-	No se pudo probar por problemas de instalación	El paquete de instalación no es de los más livianos.
Apache2	Se encuentra en el repositorio oficial y se puede instalar mediante apt	Si	Mediante interfaz gráfica. Muy simple	Autostart muy fácil de configurar con systemctl	No. Posee interfaz gráfica y muchas funcionalidades que no se usan
Lighttpd	Se encuentra en el repositorio oficial y se puede instalar mediante apt	Si	Mediante archivos que se encuentran en /etc. Muy simple.	Si. Muy simple configurar el autostart en el booteo (systemctl)	Si. Muy liviano.
Thttpd	Se instala mediante el código fuente. Usando wget	Si	Mediante archivos	No se encontró suficiente información. Falló la instalación mediante make install.	Si. Muy liviano.
Cherokee	Se instala mediante el código fuente con git clone	Si	Mediante archivos	No se lo pudo inicializar. El archivo de configuración no no se encuentra en /etc como indica la web.	Si. Es liviano
Monkey	Se instala mediante el código fuente. Usando wget	Si	Mediante archivos aunque no se lo encontró muy intuitivo	Problemas para configurar, no se encontró suficiente información	Es liviano.

Como puede verse en la mayoría de las opciones surgieron problemas de instalación o post-instalación al momento de configurar el servidor.
Nginx al no poder instalarse y no contar con CGI no se profundizó en intentar solventar el inconveniente.
Monkey se pudo instalar fácilmente mediante los pasos que indica la página oficial, pero no se logró configurarlo para el propósito de este práctico y no se encontró demasiada información además de la página web. La configuración de archivos no es tan sencilla o intuitiva, se cuenta con muchos archivos de configuración.
Thttpd parece un servidor web un poco abandonado en su desarrollo y el proceso de instalación falló. En la página oficial es muy escasa y no se presenta de manera clara.
Cherokee lanzó una serie de errores en la instalación que pudo hacer que no se creara la carpeta que, según la documentación, debería estar en el directorio **\etc**.
Los servidores Apache2 y Lighttpd funcionaron sin inconveniente alguno y su proceso de instalación es muy sencillo, simplemente mediante “sudo apt install” ya se los consigue instalar. Para el autoinicio, de nuevo, es muy sencillo mediante “systemctl”.
El proceso de configuración de apache2 es extremadamente fácil por que cuenta con una interfaz gráfica, pero considerando que nuestro hardware es una Raspberry PI y que configurar mediante “realvnc” o con un monitor y teclado no suele ser la opción más “técnica” se optó por Lighttpd que

cuenta con una configuración mediante archivos, pero muy simple e intuitiva. Algunos características pueden obtenerse simplemente descomentando líneas en su archivo de configuración (como la activación de CGI). Además existe una gran comunidad en torno a este servidor web con mucha información y experiencias que no están en la página oficial.
No se experimentaron en profundidad otros servidores web livianos como los antes mencionados debido a que se consiguió muy fácilmente hacer funcionar Lighttpd como se esperaba.

8.3. File input form

En HTML se definen 8 valores posibles para el atributo TYPE, entre los que están los usados en este trabajo como checkbox, radio, submit, etc.

En POST se define adicionalmente ENCTYPE que posee 3 formas de codificar información para ser enviada.

- application/x-www-form-urlencoded
- multipart/form-data
- text/plain

La que nos interesa es multipart/form-data.

Como se está enviando información desde el navegador hacia el servidor, se debe usar el método POST, el mismo no posee restricciones de tipo, GET solo permite caracteres ASCII.

multipart/form-data es lo que permite se puedan enviar archivos y su contenido, con GET solo podríamos enviar el nombre en texto plano.

Como su nombre indica esta codificación divide el archivo en partes y envía la información por partes.

El formulario debería ser al tipo de:

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=
POST>
File to process: <INPUT NAME="userfile1" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

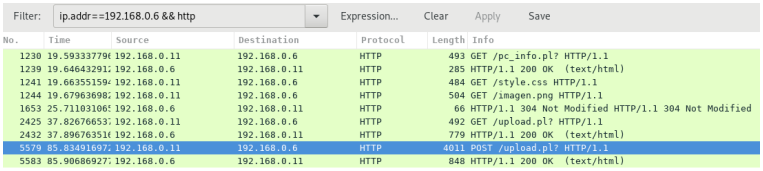
El archivo es inicialmente copiado desde el disco al navegador por STDIN y luego de presionar el boton de SUBMIT el archivo se transfiere al servidor. Desde el lado del servidor lo que se recibe es algo como:

```
  host: 'localhost:8080',
  connection: 'keep-alive',
  'content-length': '306',
  'cache-control': 'no-cache',
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit
/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36',
  'content-type': 'multipart/form-data; boundary=----
WebKitFormBoundaryvlb7BC9EAvfLB2q5',
  accept: '*/*',
  dnt: '1',
  'accept-encoding': 'gzip, deflate',
  'accept-language': 'en-US,en;q=0.8'
```

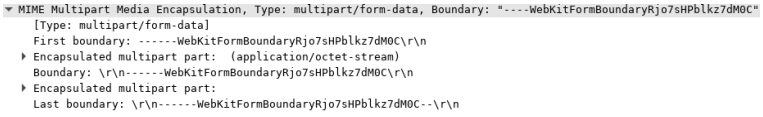
Podemos ver el tipo del contenido es "multipart/form-data" y "boundary" es lo que limita cada una de las partes del archivo que estamos enviando. Luego desde el lado del servidor se reconstruye el archivo.

A continuación mostramos algunas imagen de como se muestra la petición HTTP para nuestro caso utilizando WireShark.

En esta primera imagen observamos un único método POST en las capturas, que corresponde al script de perl encargado de la subida del archivo.



En la segunda imagen observamos el tipo "multipart/form-data" y "boundary"



Finalmente en esta última imagen podemos observar la información del archivo.