Universidad
Nacional
de Córdoba

Cátedra de Sistemas Operativos II

Trabajo Práctico Nº III

NAVARRO, Matias Alejandro

6 de junio del 2019

Índice

Introducción **3**

Propósito 3

Ámbito del Sistema 3

Definiciones, Acrónimos y Abreviaturas 3

Referencias 4

Descripción General del Documento 4

Descripción General **5**

Perspectiva del Producto 5

Funciones del Producto 5

Características de los Usuarios 5

Restricciones 5

Suposiciones y Dependencias 5

Requisitos Específicos **6**

Interfaces Externas 6

Funciones 6

Requisitos de Rendimiento 6

Restricciones de Diseño 7

Diseño de solución **7**

Implementación y Resultados **10**

Información del Sistema (System Info) 11

Formulario 12

Módulos 13

Conclusiones **16**

Apéndices **17**

Introducción

En la intersección de Software, Hardware y Comunicaciones nos podemos en contrar a los Sistemas Embebidos. Los mismos son sistemas que, si bien su definición varía con la literatura, se pueden definir como computadoras de uso específico, es decir, computadoras con requerimientos de hardware, software y comunicaciones bien definidos. Es por esto la importancia que poseen este tipo de sistemas para los Ingenieros en Computación, y que da origen al presente práctico.

En el presente Informe se mostrará el proceso de diseño, desarrollo y documentación de un software que mediante el uso de la interfaz CGI y un webserver liviano instalado en una placa embebida, permita conectarse a la misma, acceder a información del sistema, controlar, hacer una consulta a Amazon, y gestionar los módulos del Kernel en la placa.

Propósito

El propósito de este Trabajo Práctico es aprender a utilizar la interfaz CGI, de la mano de scripts escritos en Perl, o programas compilados en C, que permiten la comunicación entre el cliente y el servidor web, de manera que este último ejecute los scripts que permiten procesar archivos, para luego mostrar los resultados nuevamente en la página web.

Ámbito del Sistema

El sistema se conforma de dos partes: el webserver, que se ejecuta en una placa Raspberry PI 3B+, el cliente, que puede tomar lugar en la computadora del usuario, un celular, tablet, o cualquier dispositivo con acceso a la red.
El cliente se comunica con el server mediante la página web, e interactúa con él mediante la interfaz gráfica brindada.

Definiciones, Acrónimos y Abreviaturas

- CGI: Common Gateway Interface. interfaz de entrada común, permite a un cliente (navegador web) solicita datos de un programa ejecutado en un servidor web.
- Webserver: servidor web.
- AWS: (Amazon Web Services) es una colección de servicios de computación en la nube pública que en conjunto forman una plataforma de computación en la nube.

Referencias

1. Michael Kerrisk, The Linux Programming Interface, 2010, Adison-Wesley
2. https://registry.opendata.aws/noaa-goes/
3. http://people.cs.pitt.edu/ jmisurda/teaching/cs449/valerie-henson-device-drivers-hello.pdf
4. http://www.oreilly.com/openbook/linuxdrive3/book
5. http://en.wikipedia.org/wiki/Comparison_of_lightweight_web_servers
6. http://perldoc.perl.org/CGI.htm

Descripción General del Documento

El propósito de este documento es proporcionar a quien lo lea un entendimiento básico de los requisitos y alcance de este proyecto así como los pasos que se tomaron para el diseño e implementación al proyecto propuesto.

Descripción General

Perspectiva del Producto

El producto solicitado es un sistema que gestione la conexión con una plataforma servidor, para acceder a los datos específicos del sistema, así como también poder realizar consultas a Amazon y cargar módulos ya compilados al dicho SO.

Funciones del Producto

Las funciones que ofrece el servidor web (alojado en el sistema embebido) son: ejecución de comandos UNIX en una terminal por HTML, obtencion de informacion del sistema, llenar un formulario con el año y el día (en calendario Juliano) y que la consulta devuelva una lista del archivos disponibles de GOES16 en AWS e información de los módulos de kernel instalados en el sistema y posibilidad de instalar un nuevo módulo simple (Hello World).

Características de los Usuarios

El usuario del programa solo necesita tener un conocimiento básico de la navegación por internet y conocer la dirección IP del servidor web para poder utilizarlo. Luego, todas las opciones disponibles se presentan en pantalla con una interfaz simple y fácil de utilizar.

Restricciones

El usuario que interactúa con el web server solo tiene acceso a ciertos permisos del sistema operativo, protegiendo así al sistema.

Suposiciones y Dependencias

- El sistema embebido debe ser tipo Raspberry Pi o similar, y poseer MMU.
- Para probar el servidor web, se necesita tener instalado un navegador web.
- Existe conectividad entre el servidor y el cliente, para poder realizar la comunicación.

Requisitos Específicos

Interfaces Externas

- Para interactuar con el programa, se utiliza el navegador web, y la página que muestra el mismo. Por este medio se muestran resultados al cliente, y se introducen comandos dirigidos al servidor, quien muestra nuevamente por la página los resultados de la instrucción recibida.

- Lighttpd para configurar el webserver.

Funciones

El producto ofrece las siguientes funciones:

- Información del sistema: al ingresar a esta sección el usuario se encuentra con determinada información del sistema como lo son el Hostname, Modelo del CPU, Uso del CPU, Memoria RAM, Hora y Fecha, Uptime y Kernel de Linux.

- Archivos GOES16 disponibles: a partir de un formulario donde el usuario ingresa el año y el día juliano, se obtiene un listado de los archivos disponibles de GOES16 en AWS. Este listado que se obtiene se filtra para que sea del producto BI-L2-CMIPF, canal 13.

- Módulos: permite ver la lista de los módulos de kernel que están instalados. Además, permite cargar un módulo, instalarlo y eliminarlo.

Requisitos de Rendimiento

El proyecto debe funcionar en una placa embebida, por lo que es necesario el uso de un webserver liviano, y no uno con características que no van a usarse, como Apache. Por este motivo se eligió Lighttpd. Además, el uso de la interfaz CGI implica que el procesamiento se realiza en el servidor, lo que reafirma la necesidad de utilizar una aplicación lo más liviana posible.

Restricciones de Diseño

- El sistema operativo a instalar debe ser tipo GNU/Linux.
- El sistema embebido debe ser tipo Raspberry Pi o similar y debe poseer MMU.
- El webserver se debe ejecutar automáticamente en el inicio del sistema embebido.
- La interfaz web debe implementarse con HTML.
- Las aplicaciones deben programarse en CGI (Perl o C).
- Se debe ofrecer una forma de insertar/remover un nuevo módulo al sistema embebido.
- Se debe controlar que el archivo subido sea válido (.ko).
- Se debe utilizar Cppcheck y compilar con las flags -Werror -Wall y -pedantic.

Diseño de solución

Para el diseño de la solución se comenzó por elegir que SO instalar sobre el sistema embebido para que pueda correr el servidor.

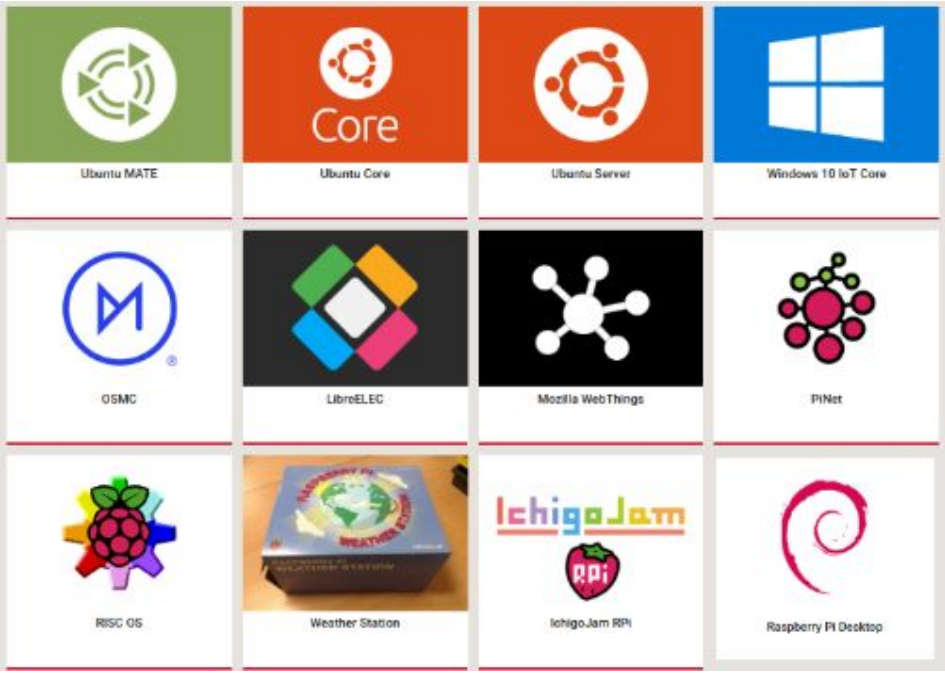

Figura 1: Sistemas Operativos para Raspberry Pi

Desde la página oficial de Raspberry se encuentran disponibles los SO vistos en la anterior figura, pero desde terceros se encuentra otra gran variedad.
El SO que se eligió para llevar a cabo dicho proyecto fue Ubuntu Mate por su simplicidad y comodidad para trabajar.

Luego, se realizó una comparativa entre los web servers disponibles y se procedió a tomar una decisión:

Figura 2: Web Servers

Apache: es el web server mas utilizado. Para los sitios web pequeños y medianos, Apache tiene varias ventajas sobre Nginx, como su fácil configuración, muchos módulos y un entorno amigable para principiantes. Pero es un web server muy pesado. Tiene soporte para CGI.

Nginx: usa menos memoria que Apache y puede manejar aproximadamente cuatro veces más solicitudes por segundo, esto último no es muy relevante en nuestro ambiente del problema dado que es una página web pequeña. Es más difícil de instalar y configurar que Apache. No tiene soporte para CGI.

Monkey server: trabaja con CGI pero se necesita compilar el servidor usted mismo, para exprimir todos los detalles y terminar con un servidor web liviano y rápido.

Lighttpd: se eligió este web server por su bajo consumo de memoria y su muy facil instalacion y configuracion (con solo un comando el webserver queda corriendo). Tiene soporte para CGI.

Por lo tanto, se procedió a instalar y configurar el web server elegido en el sistema embebido mediante los siguientes comandos:

```
sudo apt-get update

sudo apt-get install lighttpd
```

Una vez instalado, se procede a modificar el archivo de configuración del web server *lighttpd.config*. Y se inicia lighttpd con el comando *sudo /etc/init.d/lighttpd start*.

Para verificar que el web server está en funcionamiento, se navega a la IP local de la Raspberry desde cualquier navegador. Para saber la IP se utilizó el comando

```
hostname -I
```

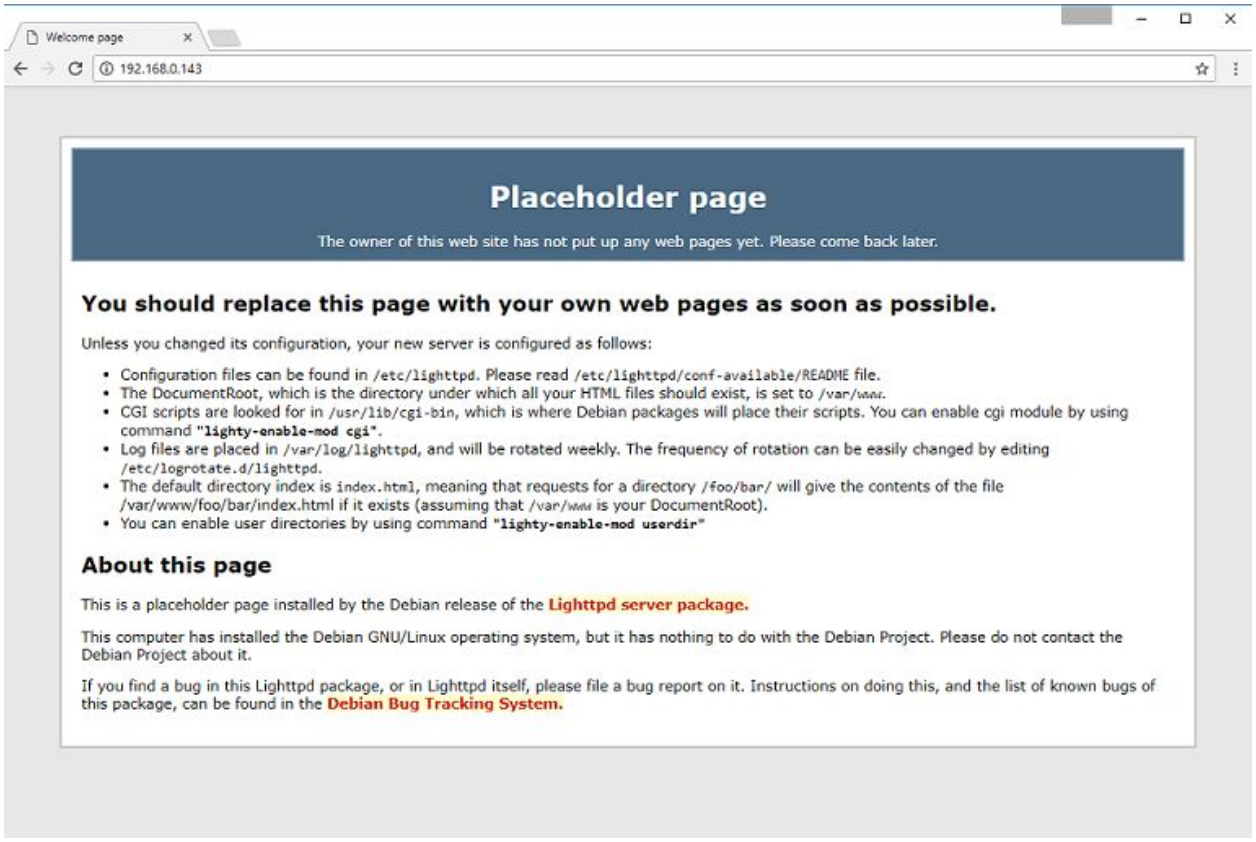

Figura 3: Pantalla de bienvenida de Lighttpd

Ahora podemos personalizar esta página, localizada en /var/www/html en nuestro sistema embebido, para editarla y de esta forma obtener nuestro propio diseño. Para ello se utilizó un template de Bootstrap y luego se modificó acorde a las características que posee la web.

Implementación y Resultados

Una vez que se modificó el template antes mencionado, los resultados fueron los siguientes:

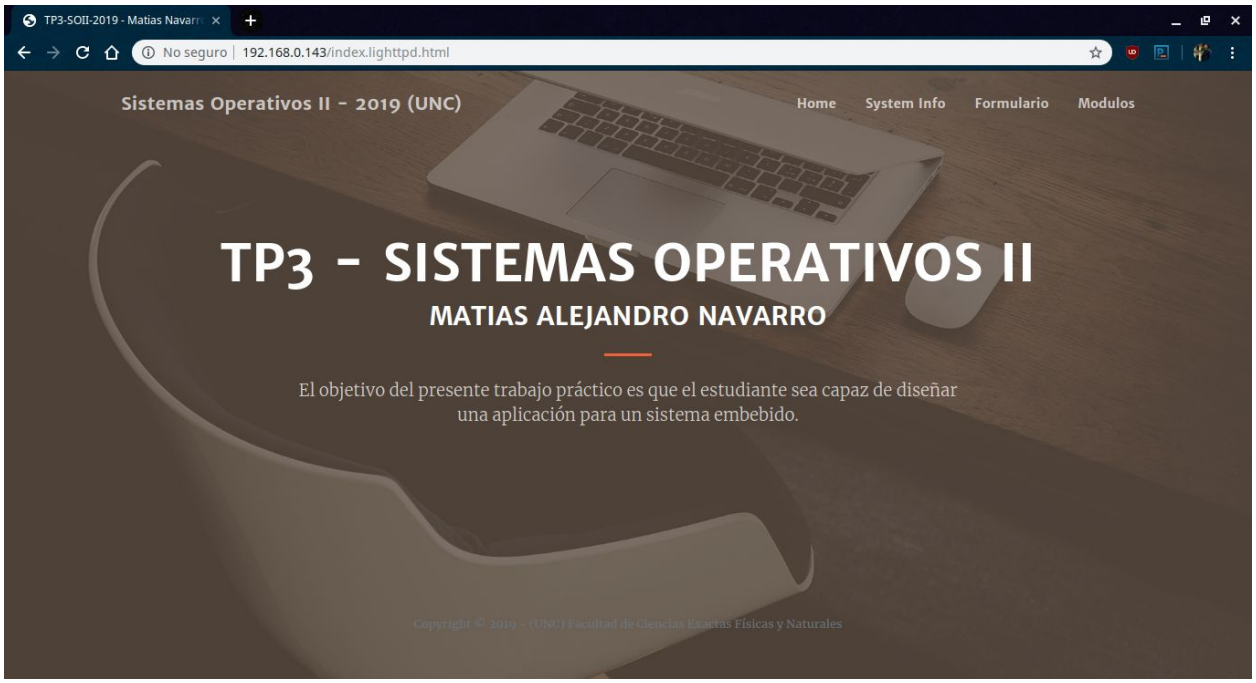

Figura 4: pantalla inicial de la página web

En la *Figura 4* se puede observar la pantalla de inicio de la página web. Con una pantalla de bienvenida y un panel (ubicado arriba a la derecha) donde se pueden acceder a las distintas partes de la página mostradas a continuación.

Información del Sistema (System Info)

Como se mencionó anteriormente esta parte de la web es la encargada de mostrar las distintas características del sistema.

Figura 4: Información del Sistema

La información del sistema es mostrada mediante un script CGI. Al presionar en la pestaña System Info, el servidor comienza a ejecutar el script ksamp.cgi, el cual busca información en /proc, lo guarda en una estructura de datos, y al finalizar imprime código HTML para que el navegador le muestre al cliente el resultado. En medio de este código HTML se encuentran los datos anteriores. Para obtener la fecha y hora actual se utilizó la estructura tm que brinda Linux, y la función localtime.

También se sigue manteniendo arriba, la barra de navegación para poder acceder a las distintas partes de la página.

Formulario

El cual permite definir una fecha (año y día DOI) que al enviarlo retorne una lista de los archivos disponibles de GOES 16 en AWS (producto de ABI-L2-CMIPF, canal 13) los cuales se hacen visibles en la sección correspondiente:

Figura 5: Formulario

Para realizar la consulta se instaló: **awscli**. La interfaz de línea de comandos (CLI) es una herramienta unificada para administrar los productos de AWS.

- Se leen los parámetros ingresados en la página web hasta detectar “=” una vez detectado, mediante la función strcat () y strtok() se obtienen el año y el dia.
- Se genera el path a ejecutar y se agrega la etiqueta --no-sign-request indicando que sea anónima para que de esta manera no sea necesario crear una cuenta.
- Al path anterior se le agrega --recursive indicando que el año y dia sea independiente de la hora.
- La ejecución del path se imprime por último en la página web.
- Método get y método post: El método **GET** se utiliza para obtener información del servidor. Traer datos que están almacenadas en el servidor, ya sea una base de datos o archivo al cliente. El método **POST** en cambio es enviar información (envío de un formulario) desde el cliente para que sea procesada y actualice o agregue información en el servidor, como sería la carga o actualización en si de un artículo.

Módulos

Listar los módulos: es un programa simple en C que hace una llamada con system al comando de UNIX lsmod.

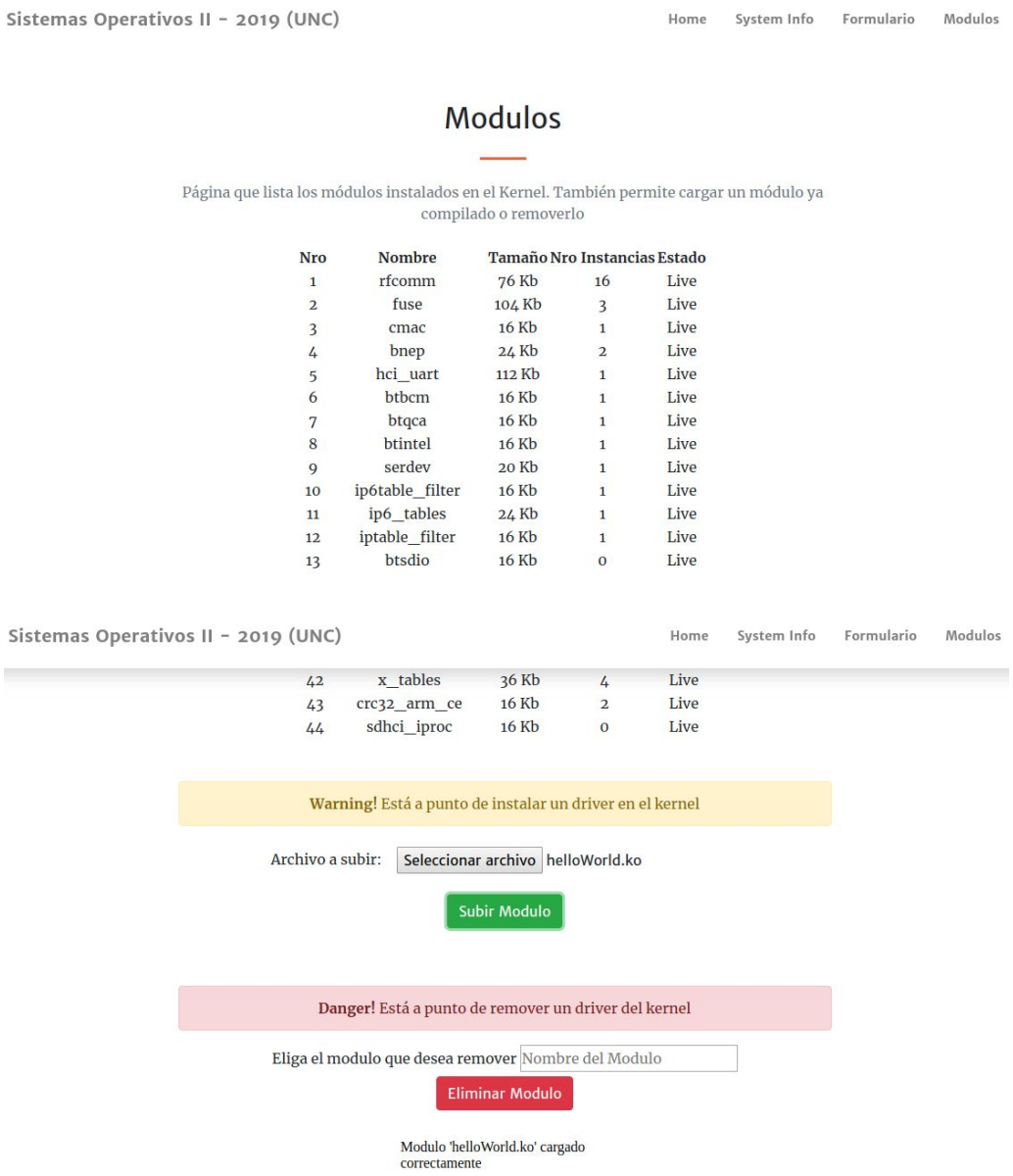

Figura 6: Módulos

Desarrollo del módulo Hello World: el cual tiene dos funciones hello_init() y hello_exit(), la primera imprime “Hola Mundo” y la segundo “Adios mundo”. Solo hace falta importar las librerías del kernel de linux y usar los prototipos init_module y cleanup_modulet. La impresión se realiza con la función `printk` y con el uso de la constante KERN_INFO, que indica que los mensajes no irán a la terminal sino al log de mensajes del SO, estos mensajes se pueden ver mediante el comando `dmseg`.

Figura 7: Eliminar módulo

Carga de un nuevo módulo:

- Se limita a 5MB el tamaño maximo del archivo.
- Se crea una lista de nombres seguros para el nombre del archivo, evitando que contenga caracteres inseguros.
- Se especifica el path donde se va guardar el archivo.
- Se crea un objeto CGI y se asigna a la variable ***$query***, esto permite usar los metodos de la libreria CGI.pm
- Leemos el nombre del archivo de subida y lo guardamos en la variable ***$filename***
- Mediante la función qr se obtiene la extensión del archivo que se almacena en la variable ***$extension***. Esto solo verifica que sea “.ko”; en caso de que se suba cualquier archivo que se le haya cambiado el nombre y puesto la extensión .ko la realiza el mismo programa insmod.
- Se verifica que cumpla con los requisitos de seguridad (caracteres válidos), se eliminan

los espacios cambiandolos por guiones bajos y finalmente se deja solo el nombre del archivo con su extensión.

- Se usa el metodo upload de la libreria CGI.pm y se guarda el archivo temporal creado por la librería en la variable ***$upload_filehandle***.
- Se lee el contenido del archivo y se guarda en un nuevo archivo en nuestro propio directorio con la función uploadfile dentro de un bucle while, escribiendo el archivo en modo binario.
- Se usa la función system de Perl para ejecutar el comando sudo ***insmod*** para cargar el módulo.
- La salida de la ejecución anterior se almacena en una variable para su posterior verificación. En caso de no producir errores se imprime que el módulo se cargó con éxito.
- Para el caso de remover el archivo del sistema se realizan pasos similares a excepción de los últimos. Y vale aclarar que se ejecuta la función ***rmmod***.

Para llevar a cabo lo anterior se tuvieron que realizar algunas modificaciones en los permisos del usuario www-data (usuario dueño del archivo subido). Este usuario aparece en el archivo /etc/passwd y se realizaron los siguientes cambios:

- Se elimino la contraseña con el comando: sudo passwd -d www-data
- Se agrego la siguiente información al archivo sudoers para que el usuario pueda ejecutar solamente los comandos insmod,rmmod y lsmod en modo sudo.

Cmnd_Alias KERNMOD= /sbin/insmod, /sbin/rmmod, /sbin/lsmod, /bin/dmesg

www-data ALL = KERNMOD

www-data: indica el nombre del usuario al que se le asigna el permiso

ALL: indica que la regla se aplica a cualquier host.

KERNMOD : indica los tres comandos que se permite ejecutar.

Para lograr el funcionamiento deseado en este punto se tuvieron que instalar los headers para realizar la compilación del módulo del kernel (apt install raspberrypi-kernel-headers) y a demás se instaló la librería para poder ejecutar Perl (sudo apt instal libcgi-session-perl).

Conclusiones

Al finalizar este trabajo se pudo aprender los beneficios de utilizar un sistema embebido y de esta manera a mentalizar que cosas implementar dependiendo de los recursos disponibles, es decir, verificar los pro y contras de las cosas que pueden utilizarse para lograr realizar la tarea específica que llevará a cabo el embebido.