Trabajo práctico Nº III - Sistemas Embebidos

CÁTEDRA DE SISTEMAS OPERATIVOS II

Casabella Martin, 39694763
martin.casabella@alumnos.unc.edu.ar

8 de junio de 2019

Índice

Introducción

En la interseccion de Software, Hardware y Comunicaciones nos podemos encontrar a los Sistemas Embebidos. Los mismos son sistemas que, si bien su definicion varıa con la literatura, se pueden definir como computadoras de uso especifico, es decir, computadoras con requerimientos de hardware, software y comunicaciones bien definidos.

Preparación de entorno

Como prerrequisito antes de iniciar con el trabajo es necesario la selección de la imagen de linux que utilizaremos en la Raspberry PI 3 Model B (de ahora en adelante rpi), cómo trabajar con ella.

Para la generación de las pruebas trabajamos con la PC local, Arch Linux / Kernel 5.1.5-1 64bits (X86_64) que tiene distinta configuración del kernel que otras distribuciones como las basadas en debian.

Elección del sistema operativo

Motivos por los cuales se opto por instalar arch:

- **Principio KISS:** Nos montamos el sistema como queremos instalando únicamente lo que necesitemos.
- **Carácter rolling release:** Nos evitamos la reinstalación de la distribución ya que no se congelan nuevas versiones.
- **Gestor de paquetes Pacman:** El gestor de paquetes Pacman es un gestor bastante rápido.
- **Yay:** Esta herramienta nos permite usar el repositorio AUR evitando a veces tener que instalar archivos .tar.gz.
- **Wiki:** La Wiki de Arch Linux es bastante extensa, pero por contra no está traducida a todos los idiomas

Como contras, se tiene que alser rolling release puede causar problemas con algún paquete, aunque Arch Linux es una de las distros más estables, y la instalación de periféricos como impresoras puede ser tediosa en algunos casos.

De todas maneras, en lo que concierne a este trabajo practico se opto principalmente porque es un linux extremadamente liviano por tener la filosofía Arch Way, mejor resumido por el acrónimo KISS.

Configuración Raspberry

En el servidor (Raspberry Pi 3 Model B), luego de instalar el Sistema Operativo, se instalaron las siguientes herramientas:

```
1 pacman -Syu  # update sistema
2 pacman -S --needed base-devel #herramientas basicas de utilidad
3 pacman -S vim git p7zip nmap nload community/mc
4 pacman -S wget community/sysstat
```

```
pacman -S community/perl-cgi #modulo cgi de PERL
pacman -S community/aws-cli #AWS


vim /etc/ssh/sshd_config  #cfg ssh


systemctrl enable sshd # Restauramos el servicio
systemctrl restart sshd # nos aseguramos que este corriendo
```

A su vez, los siguientes modulos de Perl son necesarios:

```
sudo cpanm Linux::SysInfo #info sistema
sudo cpanm Linux::Cpuinfo #info procesador
sudo cpanm File::basename #manejo de archivos
```

Elección WebServer

Se tomaron en cuenta las opciones más estables, seguras y de mejor rendimiento: apache, nginx y lighttpd.

Se prioriza un servidor liviano ya que va a correr en el embebido simplemente para esta aplicación. Ademas, se prioriza que *no tenga características que no serán utilizadas* (como Apache), y Lighttpd fue electo.

Lighttpd es un software escrito en C por Jan Kneschke, se distribuye bajo la licencia BSD y está disponible para Unix y Linux. na de las características del servidor web es que consume realmente pocos recursos a nivel de RAM y CPU, haciéndolo especialmente útil para VPS o Dedicados de bajos recursos, además de que es ideal para balancear cargas por RRDNS.

Soporta comunicación e integración con FastCGI, SCGI y CGI, por lo que es capaz de servir requests de páginas hechas en cualquier lenguaje de programación.

Nginx - Apache - Lighttpd

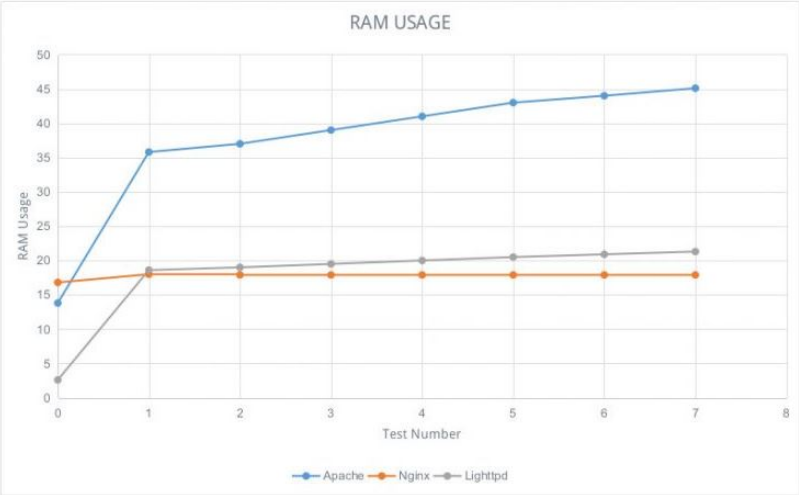

Figura 1: Uso de memoria

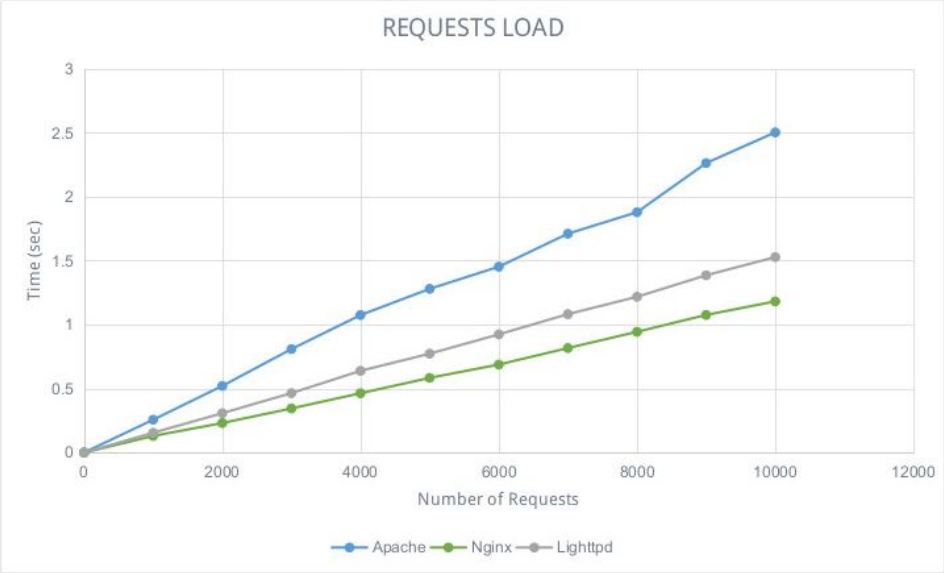

Figura 2: **Respuesta a solicitudes (performance)**

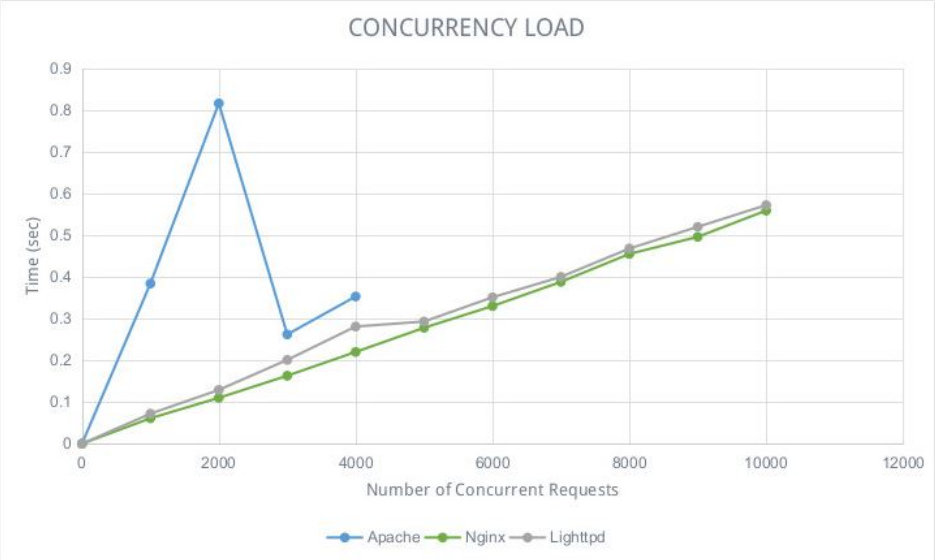

Figura 3: **Concurrencia (performance)**

Fast CGI

FastCGI es un protocolo para interconectar programas interactivos con un servidor web, donde se modifica Common Gateway Interface (CGI ó Interfaz Común de Entrada).

El principal objetivo de FastCGI es reducir la carga asociada con el hecho de interconectar el servidor web y los programas Common Gateway Interface, permitiéndole a un servidor atender más peticiones a la vez.

En vez de crear procesos nuevos por cada petición, FastCGI puede usar un solo proceso persistente que maneja cualquier petición durante su período de vida. El hecho de procesar múltiples peticiones a la vez es logrado ya sea mediante la utilización de una sola conexión con un multiplexado interno (por

ejemplo múltiples peticiones sobre una sola conexión) y/o utilizando múltiples conexiones.

FastCGI proporciona alto rendimiento y persistencia sin las limitaciones de las API específicas del servidor.Varios de esos procesos pueden existir, y eso es algo que incrementa la escalabilidad y el rendimiento. FastCGI permite también a los programas hacer que el servidor web realice ciertas operaciones sencillas, como leer un archivo antes de que la petición sea procesada.

La mencion de Fast CGI recae en el porque descartar opciones como **Nginx** , que no soporta CGI, sino que utiliza Fast CGI.

Sintesis: diferencias

- Tanto Apache como Lighttpd poseen actualmente inconvenientes como pérdidas de memoria
- Apache es el único servidor de los tres que ofrece una anulación de acceso a todo el sistema, lo que quizás sea una característica insustituible para los servicios de alojamiento compartido.
- Teniendo en cuenta los tres aspectos principales de un servidor web: características, uso de memoria y rendimiento, parece que Nginx llegó a la cima por ahora.
- Lighttpd funciona como un proceso único, mientras que Nginx funciona como un proceso maestro.
- En un sistema de archivos fragmentado funciona mejor en comparación con Nginx Lighttpd.
- La CPU utilizada por Nginx es mucho menos Lighttpd.
- Lighttpd admite IPv6 mientras se procesa la compatibilidad con Nginx IPv6.
- La función de registro de errores independiente es compatible con Nginx pero no es compatible con Lighttpd.
- Mientras que el servicio de archivos estáticos con un único servidor HTTP Lighttpd proporciona una configuración simple, pero Nginx es más difícil en el caso.
- US Lighttpd fue popular Nginx. Full Bug Tracking System es compatible con Lighttpd en Nginx tiene el sistema de seguimiento de errores de un hombre pobre.
- Los canales IRC son compatibles con Lighttpd como Nginx, pero los canales Nginx son muy silenciosos y tienen grandes retrasos para responder las preguntas de los usuarios.

Se opto por Lighttpd dado el contexo, el sistema operativo instalado, características de la aplicación (requerimientos) y uso de recursos.

Instalación Lighttpd

```
pacman –S extra/lighttpd  #instalamos el sv
systemctl enable lighttpd  #lo habilitamos
vim /etc/lighttpd/lighttpd.conf # Editamos la configuración del server para modulos CGI
systemctl status lighttpd  #lo largamos a correr
systemctl start lighttpd
systemctl status lighttpd
```

Perl scripting y HTML: How-to

Para CGI utilizamos todo por Perl Scripting debido a potencia en la manipulación de cadenas de texto, los diferentes script y la instalación de los módulos se detallan a continuación.

Por su parte, se combino el desarrollo con HTML (HyperText Markup Language) por simplicidad. El usuario opta por clickear una pestaña u otra según lo que desee, y al realizar las acciones posibles (listar módulos, listar aws, subir modulo de kernel e instalarlo, obtener información del sistema) se llama a un script hecho en Perl, que se encarga de ejecutar los comandos necesarios segun lo solicitado.

Home

La pagina principal permite al usuario elegir una pestaña u otra según lo que desee hacer.
Según su elección, se referencia a distintos archivos, encargados de obtener el resultado buscado.

index.html

```
<!DOCTYPE HTML>
<html>
<!--Pagina principal-->
<head>
  <title>Sistemas Operativos II</title>
  <meta name="description" content="website␣description" />
  <meta name="keywords" content="website␣keywords,␣website␣keywords" />
  <meta http-equiv="content-type" content="text/html;␣charset=windows-1252" />
  <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Tangerine&amp;v1" />
  <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Yanone+Kaffeesatz" />
  <link rel="stylesheet" type="text/css" href="style/style.css" />
</head>

<body>
  <div id="main">
    <div id="header">
      <div id="logo">
        <h1>Sistemas Operativos II</h1>
      </div>
      <div id="menubar">
        <ul id="menu">
          <!-- put class="current" in the li tag for the selected page - to highlight which page you're on -->
          <li class="current"><a href="index.html">Home</a></li>
          <li><a href="cgi-bin/system_info.pl">System Resources</a></li>
          <li><a href="goes.html">GOES Info</a></li>
          <li><a href="cgi-bin/modules.pl">System modules</a></li>
          <li><a href="upload_module.html">Install module</a></li>
        </ul>
      </div>
    </div>

      <div id="content">
        <!-- insert the page content here -->
        <h1>Welcome to OSII server</h1>
        <p>This simple web server provides information about GOES 16 Satellite. Clients could also request information
          about system resources, and system modules. </p>
        <p>It is also allowed to upload  a file to server, and install or remove desired module into system kernel</p>
       <p></p>
        <p></p>
        <p></p>
        <p></p>
        <p></p>
        <p></p>
        <p></p>

      </div>
    </div>
```

```
  <p></p>
  <p></p>
  <p></p>
  <p></p>
  <p></p>
  <p></p>
    <div id="footer">
      Martin Casabella</p>martin.casabella@alumnos.unc.edu.ar</p>
<!--      <img src="icons/UNClogo.jpg" style="width:250px;height:100px;" >-->
    </div>
  </div>
</body>
</html>
```

El script referencia diferentes scripts que se ejecutan según la pestaña correspondiente.

System Resources

Simplemente referencia un script en perl, encargado de ejecutarse y obtener informacion deseada del sistema segun los requerimientos. Para ello se utilizan módulos disponibles en Perl, como Linux::SysInfo, y Linux::CpuInfo, que hacen de interfaz con las llamadas del sistema operativo.

system_info.pl

```
#!/usr/bin/perl
use strict;
use warnings FATAL => 'all';
use lib qw<blib/lib blib/arch>;

use CGI::Carp qw(fatalsToBrowser);
use Linux::SysInfo qw<sysinfo>;
use Proc::CPUUsage;
use Unix::Uptime qw(:hires);


#para uso CPU
my $cp = Proc::CPUUsage->new;
#para sys info
my $si = sysinfo;

my @months = qw( Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec );
my @days = qw(Sun Mon Tue Wed Thu Fri Sat Sun);
my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime();
$year += 1900;

#Date/time
my $current_date = "$days[$wday] $mday $months[$mon] $year";
my $current_time = "$hour:$min:$sec";

#CPU usage
my $usage1 = $cp->usage*100; ## returns usage since new()
my $usage2 = $cp->usage*100; ## returns usage since last usage()

#Uptime
my $uptime = Unix::Uptime->uptime(); # 2345
# # "HiRes" mode
my $uptime_sec =Unix::Uptime->uptime_hires();

my ($load1, $load5, $load15) = Unix::Uptime->load(); # (1.0, 2.0, 0.0)


#HTML FORMAT PAGE
```

```
40 print "Content-type: text/html\n\n";
41
42 print "<html>\n";
43 print "<head>";
44 print "<title>Sistemas Operativos II</title>";
45 print "<meta name=\"description\" content=\"website description\" /> ";
46 print "<meta name=\"keywords\" content=\"website keywords, website keywords\" />";
47 print "<meta http-equiv=\"content-type\" content=\"text/html; charset=windows-1252\" />";
48 print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
       =Tangerine&amp;v1\" />";
49 print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
       =Yanone+Kaffeesatz\"/>";
50 print "<link rel=\"stylesheet\" type=\"text/css\" href=\"../style/style.css\" />";
51 print "</head>";
52
53
54 print "<body>";
55 <body bgcolor="#ff69b4" value="F" >;
56 print "<div id=\"main\">";
57 print "<div id=\"header\">";
58 print "<div id=\"logo\">";
59 print "<h1>Sistemas Operativos II</h1>";
60 print "</div>";
61 print "<div id=\"menubar\">";
62 print "<ul id=\"menu\">";
63 print "<li><a href=\"../index.html\">Home</a></li>";
64 print "<li class=\"current\"><a href=\"system_info.pl\">System Resources</a></li>";
65 print "<li><a href=\"../goes.html\">GOES Info</a></li>";
66 print "<li><a href=\"modules.pl\">System modules</a></li>";
67 print "<li><a href=\"../upload_module.html\">Install module</a></li>";
68 print "</ul>";
69 print "</div>";
70 print "</div>";
71
72
73 print "<div id=\"content\">";
74 print "<h1>System Resources</h1>";
75 print "<p>Date: $current_date        Time: $current_time";
76 print "<p><p>System information:</p>";
77 print "<p>- $_: $si->{$_}" for keys %$si;
78 print "<p>- CPU usage (last measurement):  $usage2 %</p>";
79 print "<p> -Uptime: " , $uptime_sec, " [s]</p>";
80 print "<p> -Load average for last minute: " ,$load1, </p>;
81
82 print "     <p></p>";
83 print "  <p></p>";
84 print "      <p></p>";
85 print "   <p></p>";
86
87 print "</div>";
88 print "</div>";
89 print "<p></p>";
90 print "<p></p>";
91
92 print " <div id=\"footer\">";
93 print "Martin Casabella</p>martin.casabella\@alumnos.unc.edu.ar</p>";
94 print " </div>";
95 print " </div>";
96 print " </body>";
97 print "  </html>";
```

GOES Info

Devuelta, al elegir esta pestana, se referencia primero a un archivo html, que mediante un formulario que se le brinda al usuario, obtiene la fecha deseada mediante un formulario y que al enviarlo, devuelve la lista del archivos disponibles de Goes 16 en AWS (producto ABI-L2-CMIPF, canal 13) y los muestra en la

pagina. Para ello, desde este archivo se ejecuta un nuevo script hecho en Perl, que parsea este ingreso, y ejecuta el comando aws para obtener la información deseada.

goes.html

```
<!DOCTYPE HTML>
<html>

<head>
    <title>Sistemas Operativos II</title>
    <meta name="description" content="website␣description" />
    <meta name="keywords" content="website␣keywords,␣website␣keywords" />
    <meta http-equiv="content-type" content="text/html;␣charset=windows-1252" />
    <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Tangerine&amp;v1" />
    <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Yanone+Kaffeesatz" />
    <link rel="stylesheet" type="text/css" href="style/style.css" />
</head>

<body>
<div id="main">
    <div id="header">
        <div id="logo">
            <h1>Sistemas Operativos II</h1>
        </div>
        <div id="menubar">
            <ul id="menu">
                <!-- put class="current" in the li tag for the selected page - to highlight which page you're on -->
                <li><a href="index.html">Home</a></li>
                <li><a href="cgi-bin/system_info.pl">System Resources</a></li>
                <li class="current"><a href="goes.html">GOES Info</a></li>
                <li><a href="cgi-bin/modules.pl">System modules</a></li>
                <li ><a href="upload_module.html">Install module</a></li>
            </ul>
        </div>
    </div>

    <div id="content">

        <!--   Se le pide fecha al usuario y se llama a un script de perl que se encarga de ejecutar
             el aws hacia el GOES para listar los archivos de la fecha introducida-->

        <h1>GOES 16 AWS</h1>
        <p></p>
        <p>Enter date for searching with following format:  year/day/hour</p>
        <p></p>
        <form action="cgi-bin/aws.pl" method="post" enctype="multipart/form-data">
            <input type="text" name="date">
            <p></p>
            <input type="submit" value="submit">
        </form>


    </div>

    <p></p>
    <p></p>
    <p></p>
    <p></p>
    <p></p>
    <p></p>
    <p></p>

</div>
</div>
<p></p>
<p></p>
<p></p>
<p></p>
<p></p>
<p></p>
<div id="footer">
```

```
    Martin Casabella</p>martin.casabella@alumnos.unc.edu.ar</p>
    <!--      <img src="icons/UNClogo.jpg" style="width:250px;height:100px;" >-->
</div>
</div>
</body>
</html>
```

aws.pl

```
#!/usr/bin/perl -w
use warnings;
use strict;
use CGI qw(:standard);
use CGI::Carp qw(fatalsToBrowser);
use File::Basename;
use File::chdir;
#Instancia CGI
my $query = new CGI;


#obtengo fecha como parametro del ususario
my $date = $query->param("date");
#my $date = "2019/025/01";
my @words = split '/', $date;

my $year = $words[0];
my $day = sprintf '%03s', $words[1];
my $hour = sprintf '%02s', $words[2];


#Ejecuto comando aws, y parseo su salida, para asi imprimirla
my $raw_out  = qx(/usr/bin/aws --no-sign-request s3 ls s3://noaa-goes16/ABI-L2-CMIPF/$year/$day
    /$hour/ 2>&1);

#print '/usr/bin/aws --no-sign-request s3 ls s3://noaa-goes16/ABI-L2-CMIPF/2019/001/23/ 2>&1';
my @lines = split /\n/, $raw_out;


#Muestro pagina html para seguir formato de las demas
print "Content-type: text/html\n\n";

print "<html>\n";
print "<head>";
print "<title>Sistemas Operativos II</title>";
print "<meta name=\"description\" content=\"website description\" /> ";
print "<meta name=\"keywords\" content=\"website keywords, website keywords\" />";
print "<meta http-equiv=\"content-type\" content=\"text/html; charset=windows-1252\" />";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
    =Tangerine&amp;v1\" />";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
    =Yanone+Kaffeesatz\"/>";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"../style/style.css\" />";
print "</head>";


print "<body>";
<body bgcolor="#ff69b4" value="F" >;
print "<div id=\"main\">";
print "<div id=\"header\">";
print "<div id=\"logo\">";
print "<h1>Sistemas Operativos II</h1>";
print "</div>";
print "<div id=\"menubar\">";
print "<ul id=\"menu\">";
print "<li><a href=\"../index.html\">Home</a></li>";
print "<li ><a href=\"system_info.pl\">System Resources</a></li>";
print "<li class=\"current\" ><a href=\"../goes.html\">GOES Info</a></li>";
print "<li><a href=\"modules.pl\">System modules</a></li>";
```

```
print "<li><a href=\"../upload_module.html\">Install module</a></li>";
print "</ul>";
print "</div>";
print "</div>";


print "<div id=\"content\">";
print "<h1>GOES 16 archives</h1>";

print "$year\n";
print "$day\n";
print "$hour\n";
print"<p><\p>";

# Imprimo lista de archivos en el html para que el usuario vea lo solicitado
foreach my $line (@lines) {
    my @entry = split('\s+',$line,4);
     print "$entry[0]      $entry[1]   $entry[2]       $entry[3] ";


}

print "       <p></p>";
print "    <p></p>";
print "       <p></p>";
print "     <p></p>";
print "         <p></p>";

print "</div>";
print "</div>";
print "<p></p>";
print "<p></p>";

print " <div id=\"footer\">";
print "Martin Casabella</p>martin.casabella\@alumnos.unc.edu.ar</p>";
print " </div>";
print " </div>";
print " </body>";
print "  </html>";
```

System Modules

Aqui se satisface el requerimiento de devolver en la pagina la lista con modulos instalados. Para ello, se utiliza el comando *lnsmod*. Una prueba que se realizo es probar obtener la lista, proceder a instalar el modulo (con la otra pestana) y volver a pedir la lista, verificando que se haya instalado el modulo.

module.pl

```
#!/usr/bin/perl
use strict;
use warnings FATAL => 'all';
use lib qw<blib/lib blib/arch>;
use CGI qw(:standard);


#Obtengo lista de modulos
my $out = qx(lsmod 2>&1);
#my @lines = split /\n/, $out; #guardo en array lines, lista partida en \n


print "Content-type: text/html\n\n";
```

```
14
15 print "<html>\n";
16 print "<head>";
17 print "<title>Sistemas Operativos II</title>";
18 print "<meta name=\"description\" content=\"website description\" /> ";
19 print "<meta name=\"keywords\" content=\"website keywords, website keywords\" />";
20 print "<meta http-equiv=\"content-type\" content=\"text/html; charset=windows-1252\" />";
21 print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
       =Tangerine&amp;v1\" />";
22 print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
       =Yanone+Kaffeesatz\"/>";
23 print "<link rel=\"stylesheet\" type=\"text/css\" href=\"../style/style.css\" />";
24 print "</head>";
25
26 print "<body>";
27 <body bgcolor="#ff69b4" value="F" >;
28 print "<div id=\"main\">";
29 print "<div id=\"header\">";
30 print "<div id=\"logo\">";
31 print "<h1>Sistemas Operativos II</h1>";
32 print "</div>";
33 print "<div id=\"menubar\">";
34 print "<ul id=\"menu\">";
35 ;
36
37 print "<li><a href=\"../index.html\">Home</a></li>";
38 print "<li><a href=\"system_info.pl\">System Resources</a></li>";
39 print "<li><a href=\"../goes.html\">GOES Info</a></li>";
40 print "<li class=\"current\"><a href=\"modules.pl\">System modules</a></li>";
41 print "<li><a href=\"../upload_module.html\">Install module</a></li>";
42 print "</ul>";
43 print "</div>";
44 print "</div>";
45
46 print "<div id=\"content\">";
47 print "<h1>Installed system modules</h1>";
48
49
50 #Parseo salida lsmod y la muestro en pantalla
51 my @lines = split /\n/, $out;
52
53 #Ssimplemente para obviar la primera linea de salida
54 my $first = 1;
55 foreach my $line (@lines) {
56     my @entry = split('\s+',$line,3);
57
58     if ($first==1){
59         $first = 0;
60     }else{
61         print "<p> Module:  $entry[0]   Size:  $entry[1]   Used by:  $entry[2]
       ";
62     }
63 }
64
65
66 print "</div>";
67 print "</div>";
68 print "<p></p>";
69 print "<p></p>";
70
71 print " <div id=\"footer\">";
72 print "Martin Casabella</p>martin.casabella\@alumnos.unc.edu.ar</p>";
73 print " </div>";
74 print " </div>";
75 print " </body>";
76 print "  </html>";
```

Install module

Se tienen dos instancias para proceder a instalar un modulo: verificar que el archivo sea valido, y luego enviarlo para instalarlo. Todo se realizo en otro script en Perl, donde se valida primero el archivo verificando extension .ko, y luego se envia hacia el servidor, donde se coloca en el directorio definido para las cargas *uploads*.
Una vez validado y enviado, desde el mismo script, se ejecuta el comando *insmodmodule.ko* para proceder a cargarlo. Se handlean diferentes errores en el script (mala extension, no existencia del archivo, entre otros).

upload_module.html

```
<!DOCTYPE HTML>
<html>
<!--Pagina que se muestra al seleccionar la pestana Install MOdule. Permite al usuario cargar un modulo
de kernel compilado, y enviarlo al servidor. De ser valido, se instala. Caso contrario, avisa que no es
archivo tipo modulo y no procede a cargarlo ni a instalarlo.

A su vez, se puede eliminar un modulo previamente enviado y cargado. SI no existe en la carpeta del servidor,
no hara nada. SI el modulo existe, haciendo referencia a que previamente se cargo, se remueve, y se elimina
el archivo .ko perviamente cargado.
-->

<head>
    <title>Sistemas Operativos II</title>
    <meta name="description" content="website␣description" />
    <meta name="keywords" content="website␣keywords,␣website␣keywords" />
    <meta http-equiv="content-type" content="text/html;␣charset=windows-1252" />
    <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Tangerine&amp;v1" />
    <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Yanone+Kaffeesatz" />
    <link rel="stylesheet" type="text/css" href="style/style.css" />
</head>

<body>
<div id="main">
    <div id="header">
        <div id="logo">
            <h1>Sistemas Operativos II</h1>
        </div>
        <div id="menubar">
            <ul id="menu">
                <!-- put class="current" in the li tag for the selected page - to highlight which page you're on -->
                <li><a href="index.html">Home</a></li>
                <li><a href="cgi-bin/system_info.pl">System Resources</a></li>
                <li><a href="goes.html">GOES Info</a></li>
                <li><a href="cgi-bin/modules.pl">System modules</a></li>
                <li class="current"><a href="upload_module.html">Install module</a></li>
            </ul>
        </div>
    </div>

    <div id="content">

        <!-- insert the page content here -->
        <h1>Install or remove kernel module</h1>
        <!--Con el archivo seleccionado se llama al script de perl-->
        <form action="/cgi-bin/upload_module.pl" method="post" enctype="multipart/form-data">
        <p>File to upload: <input type="file" name="module" /></p>
        <p><input class="w3-button" type="submit" name="Submit" value="␣␣Upload␣module␣" /></p>
        </form>
        <p><a class="w3-button" href="cgi-bin/rm_module.pl">Delete uploaded module</a></p>
        </div>
        <p></p>
        <p></p>
        <p></p>
        <p></p>
        <p></p>
        <p></p>
```

```
        <p></p>
    </div>
</div>
<p></p>
<p></p>
<p></p>
<p></p>
<p></p>
<p></p>
<div id="footer">
    Martin Casabella</p>martin.casabella@alumnos.unc.edu.ar</p>
    <!--      <img src="icons/UNClogo.jpg" style="width:250px;height:100px;" >-->
</div>
</div>
</body>
</html>
```

upload_module.pl

```
#!/usr/bin/perl
use strict;
use warnings FATAL => 'all';
use CGI;
use CGI::Carp qw ( fatalsToBrowser );
use File::Basename;
use File::chdir;




print "Content-type: text/html\n\n";

print "<html>\n";
print "<head>";
print "<title>Sistemas Operativos II</title>";
print "<meta name=\"description\" content=\"website description\" /> ";
print "<meta name=\"keywords\" content=\"website keywords, website keywords\" />";
print "<meta http-equiv=\"content-type\" content=\"text/html; charset=windows-1252\" />";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
    =Tangerine&amp;v1\" />";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
    =Yanone+Kaffeesatz\"/>";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"../style/style.css\" />";
print "</head>";


print "<body>";
<body bgcolor="#ff69b4" value="F" >;
print "<div id=\"main\">";
print "<div id=\"header\">";
print "<div id=\"logo\">";
print "<h1>Sistemas Operativos II</h1>";
print "</div>";
print "<div id=\"menubar\">";
print "<ul id=\"menu\">";
print "<li><a href=\"../index.html\">Home</a></li>";
print "<li ><a href=\"system_info.pl\">System Resources</a></li>";
print "<li><a href=\"../goes.html\">GOES Info</a></li>";
print "<li><a href=\"modules.pl\">System modules</a></li>";
print "<li class=\"current\" ><a href=\"../upload_module.html\">Install module</a></li>";
print "</ul>";
print "</div>";
print "</div>";


print "<div id=\"content\">";
print "<h1>Install or remove kernel module</h1>";

```

```
48
49
50 $CGI::POST_MAX = 1024 * 5000;
51
52 # my $safe_filename_characters = "a-zA-Z0-9_.-";
53 my $upload_dir = "/srv/http/uploads";
54 my $query = new CGI;
55 my $filename = $query->param("module");
56
57 #COn el parametro module recibido (y archivo) procedo a validarlo
58
59 if ( !$filename ) {
60     # print $query->header ( );
61     # print "There was a problem uploading your photo (try a smaller file).";
62     print "<p>Iput not interpreted\n\n";
63     exit;
64 }
65
66
67 #Parseo la extension
68 my ( $name, $path, $extension ) = fileparse ( $filename, '..*' );
69 $filename = $name . $extension;
70
71 $filename =~ tr/ /_/;
72 # $filename =~ s/[^$safe_filename_characters]//g;
73
74 if ($filename =~ /^(\w+[\w.-]+\.ko)$/o) {
75     $filename = $1;
76     print "<p> Checked file: [ OK ]. <p>"
77 }
78 else {
79     die "<p> Not a module file <p>";
80 }
81
82 my $upload_filehandle = $query->upload("module");
83
84 open ( UPLOADFILE, ">$upload_dir/$filename" ) or die "$!";
85 binmode UPLOADFILE;
86
87 while ( <$upload_filehandle> ){
88     print UPLOADFILE;
89 }
90
91 close UPLOADFILE;
92
93
94 chdir "../uploads";
95 print `/usr/bin/sudo /usr/bin/insmod $filename`;
96 print " Module uploaded to server and installed correctly <p>.";
97
98 print "      <p></p>";
99 print "   <p></p>";
100 print "      <p></p>";
101 print "    <p></p>";
102 print "        <p></p>";
103
104 print "</div>";
105 print "</div>";
106 print "<p></p>";
107 print "<p></p>";
108
109 print " <div id=\"footer\">";
110 print "Martin Casabella</p>martin.casabella\@alumnos.unc.edu.ar</p>";
111 print " </div>";
112 print " </div>";
113 print " </body>";
114 print "  </html>";
```

rm_module.pl

Da como supuesto que el modulo a remover, es uno previamente cargado por el usuario, por lo que debe existir como requisito, el archivo *module.ko* en la carpeta uploads.

Si este no existe en la misma, no se efectua ninguna operacion, ya que puede recaer en la descarga de algun modulo existente en el servidor. Si existe, se ejecuta *rmmodmodule.ko* y luego se borra el archivo *.ko* del directorio uploads. Nuevamente, se realizo todo en Perl.

```
#!/usr/bin/perl
use strict;
use warnings FATAL => 'all';
use CGI;
use CGI::Carp qw ( fatalsToBrowser );
use File::Basename;
use File::chdir;




print "Content-type: text/html\n\n";

print "<html>\n";
print "<head>";
print "<title>Sistemas Operativos II</title>";
print "<meta name=\"description\" content=\"website description\" /> ";
print "<meta name=\"keywords\" content=\"website keywords, website keywords\" />";
print "<meta http-equiv=\"content-type\" content=\"text/html; charset=windows-1252\" />";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
    =Tangerine&amp;v1\" />";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"http://fonts.googleapis.com/css?family
    =Yanone+Kaffeesatz\"/>";
print "<link rel=\"stylesheet\" type=\"text/css\" href=\"../style/style.css\" />";
print "</head>";


print "<body>";
<body bgcolor="#ff69b4" value="F" >;
print "<div id=\"main\">";
print "<div id=\"header\">";
print "<div id=\"logo\">";
print "<h1>Sistemas Operativos II</h1>";
print "</div>";
print "<div id=\"menubar\">";
print "<ul id=\"menu\">";
print "<li><a href=\"../index.html\">Home</a></li>";
print "<li ><a href=\"system_info.pl\">System Resources</a></li>";
print "<li><a href=\"../goes.html\">GOES Info</a></li>";
print "<li><a href=\"modules.pl\">System modules</a></li>";
print "<li class=\"current\" ><a href=\"../upload_module.html\">Install module</a></li>";
print "</ul>";
print "</div>";
print "</div>";


print "<div id=\"content\">";
print "<h1>Install or remove kernel module</h1>";
print "<p> Module removed <p>";
#Cambio a directorio de carga del sv
chdir "../uploads";
#Defino path
my $dirname = "/srv/http/uploads";
#abro directorio y handlero error
opendir ( DIR, $dirname ) || die "Error in opening dir $dirname\n";
#busco .ko files, para removerlas
my @files = grep(/\.ko$/,readdir(DIR));
```

```
56 closedir(DIR);
57 #Si hay alguna, significa que el ususario previamente envio el modulo (lo cargo e instalo).
58 #Lo remuevo, y borro el .ko del directorio del sv
59 foreach my $file (@files) {
60       print `/usr/bin/sudo /usr/bin/rmmod $file `;
61       print `/usr/bin/sudo /usr/bin/rm $file `;
62 }
63
64
65
66
67 print "       <p></p>";
68 print "    <p></p>";
69 print "       <p></p>";
70 print "     <p></p>";
71 print "         <p></p>";
72
73 print "</div>";
74 print "</div>";
75 print "<p></p>";
76 print "<p></p>";
77
78 print " <div id=\"footer\">";
79 print "Martin Casabella</p>martin.casabella\@alumnos.unc.edu.ar</p>";
80 print " </div>";
81 print " </div>";
82 print " </body>";
83 print "  </html>";
```

POST, GET y CGI

Introduciendo, se sabe que GET se utiliza para solicitar datos de un recurso específico, mientras que POST se utiliza para enviar datos a un servidor para crear / actualizar un recurso.

Referenciando CGI,La principal diferencia entre estos métodos es la forma en que los datos del formulario se pasan al programa CGI.

- Si se utiliza el método GET, la cadena de consulta simplemente se agrega a la URL del programa cuando el cliente envía la solicitud al servidor. Se puede acceder a esta cadena de consulta utilizando la variable de entorno QUERY_STRING.
- Para obtener datos enviados por el método POST, el programa CGI lee desde *stdin*

El servidor necesita una forma de saber qué URL se asignan a los scripts y qué URL solo se asignan a los archivos HTML normales. Para CGI, esto suele hacerse mediante la creación de directorios CGI en el servidor.

Esto se hace en la configuración del servidor y le dice al servidor que todos los archivos en un directorio de nivel superior en particular son scripts CGI (ubicados en algún lugar del disco) que se ejecutarán cuando se solicite. (El directorio predeterminado suele ser / cgi-bin /, por lo que se puede decir que las URL así: http://www.varsity.edu/cgi-bin/search apuntan a una secuencia de comandos CGI. Tenga en cuenta que el directorio puede llamarse de cualquier manera .)

Básicamente, CGI funciona así: se envía una URL al servidor, que usa CGI para ejecutar un programa. El servidor pasa la entrada al programa y la salida del programa de vuelta al emisor. CGI actúa como una "puerta de enlace.entre el servidor y el programa escrito.

El principio básico de Common Gateway Interface (CGI) es que un servidor web pasa la información de solicitud del cliente a los programas CGI en las variables de entorno del sistema (y en algunos casos

a través de argumentos de línea de comando o entrada estándar) y toda la salida estándar de los programas CGI se devuelve a la Web (clientela).

Cuando las variables de entorno han sido establecidas por el servidor HTTP, inicia el programa CGI. Depende de este programa CGI averiguar dónde obtener la información necesaria para cumplir con la solicitud.

CGI: procesando entradas

Cuando las variables de entorno han sido establecidas por el servidor HTTP, inicia el programa CGI. (Para obtener una lista completa de las variables de entorno establecidas por el servidor HTTP, consulte las variables de entorno establecidas por el servidor HTTP). Luego, depende de este programa CGI averiguar dónde obtener la información necesaria para cumplir con la solicitud.

Procesando solicitud

Procesar la solicitud es la segunda etapa de un programa CGI. En esta etapa, el programa toma los datos analizados y realiza la acción apropiada

Respuesta

Cuando el programa CGI ha terminado de procesarse, debe enviar su resultado al servidor HTTP que invocó el programa. Al hacerlo, la salida se envía indirectamente al cliente que inicialmente solicitó la información.

Debido a que el programa CGI emite su resultado a través de STDOUT, el servidor HTTP debe leer la información desde allí e interpretar qué hacer.

Un programa CGI escribe un encabezado CGI que es seguido por un cuerpo de entidad en la salida estándar. El encabezado CGI es la información que describe los datos en el cuerpo de la entidad. El cuerpo de la entidad son los datos que el servidor envía al cliente.

Implementacion y resultados

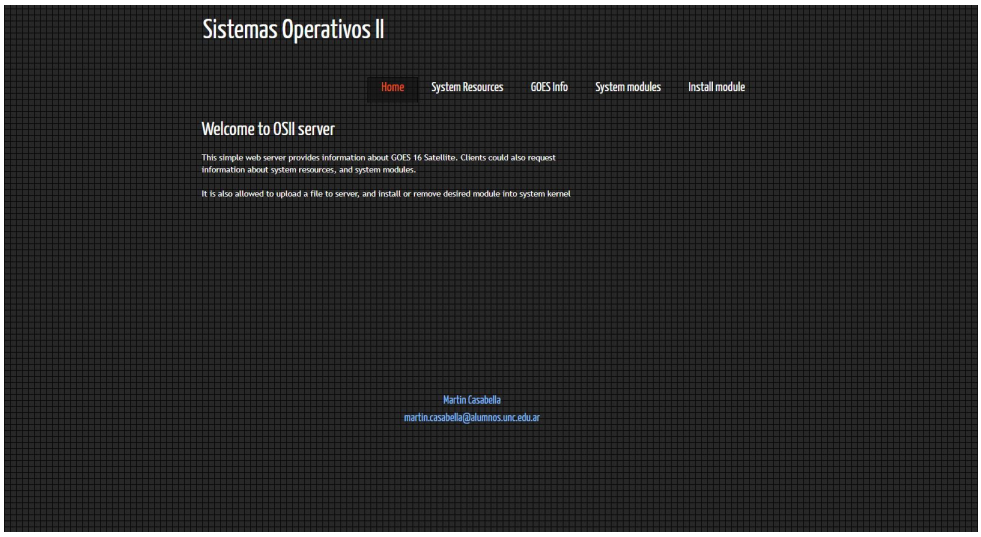

Figura 4: Pagina principal

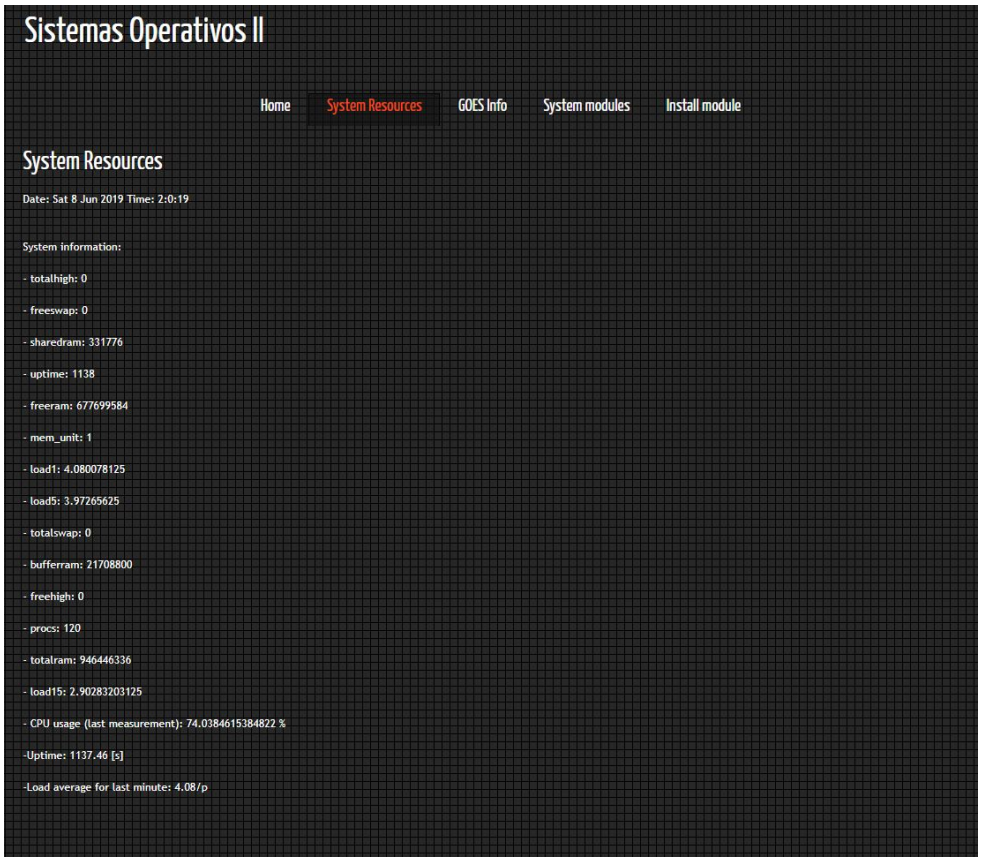

Figura 5: Pestaña que lista recursos del sistema

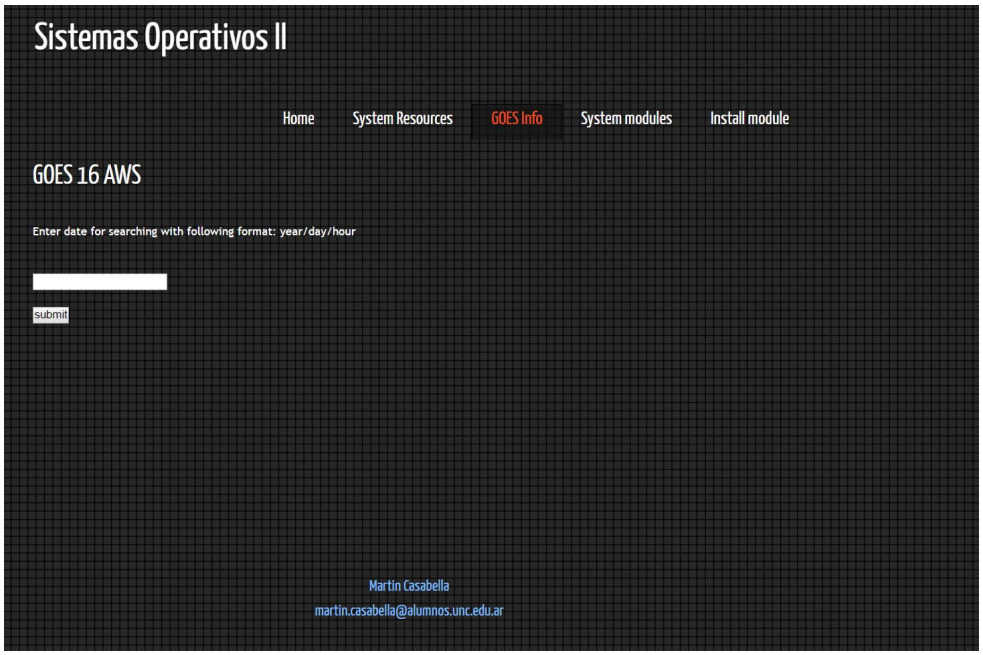

Figura 6: Pestaña para obtener información del GOES

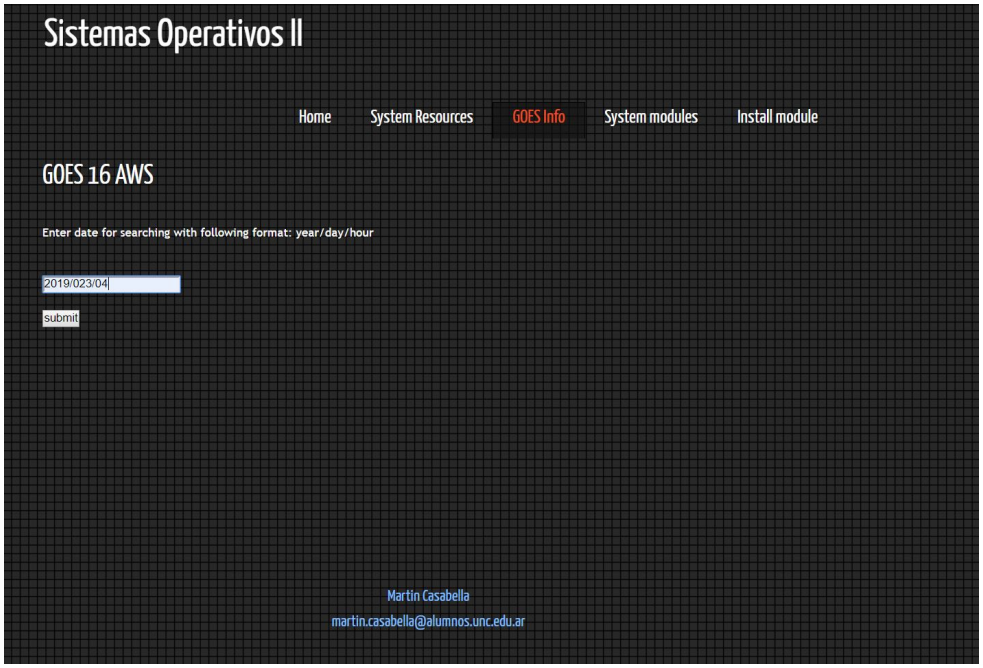

Figura 7: Entrada para testeo

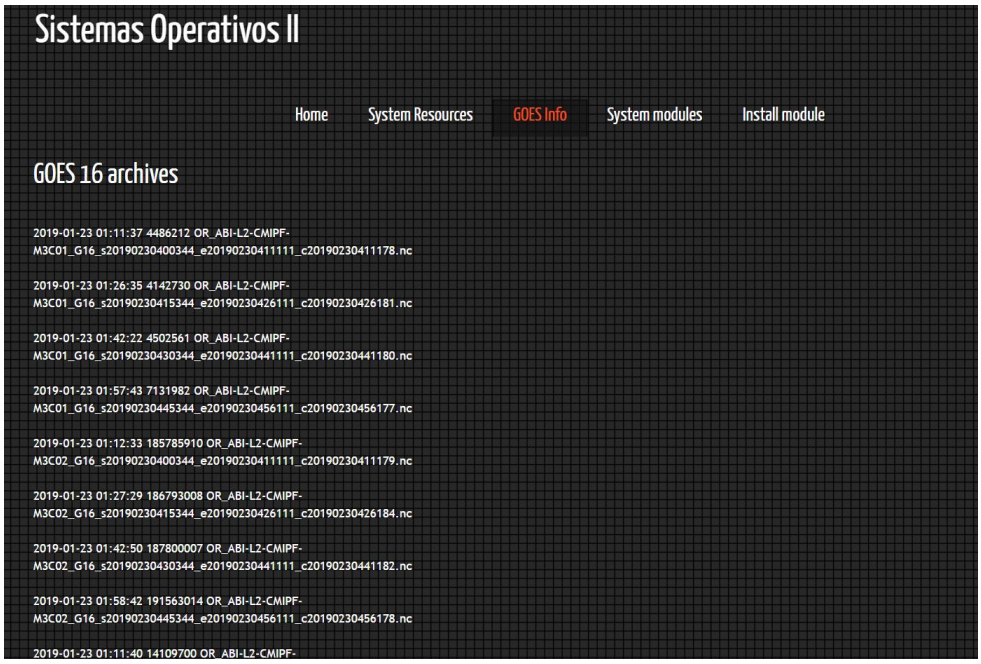

Figura 8: **Devolución**

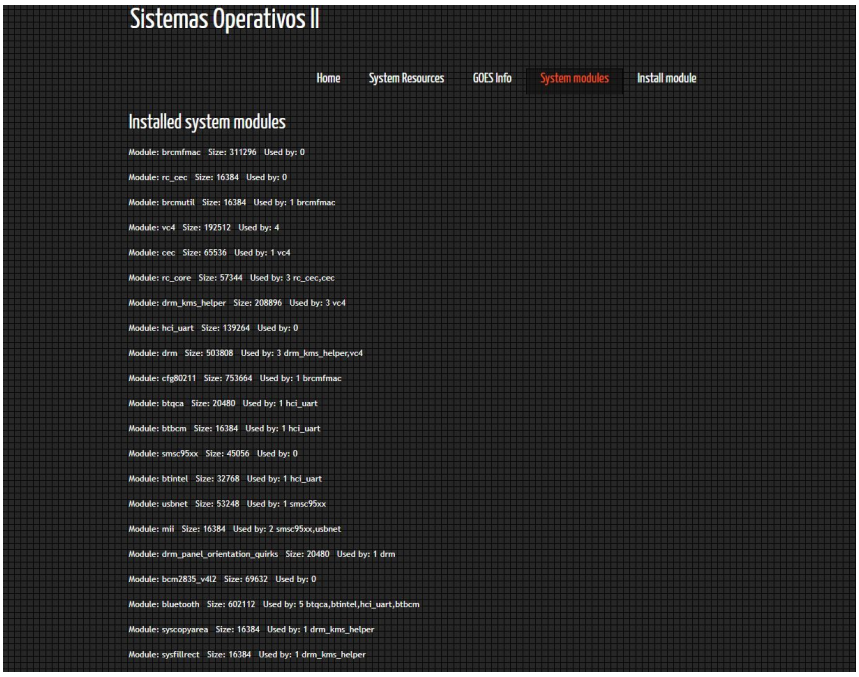

Figura 9: **Pestaña que lista modulos del kernel instalados**

Figura 10: **Pestaña que permite elegir archivo a cargar**

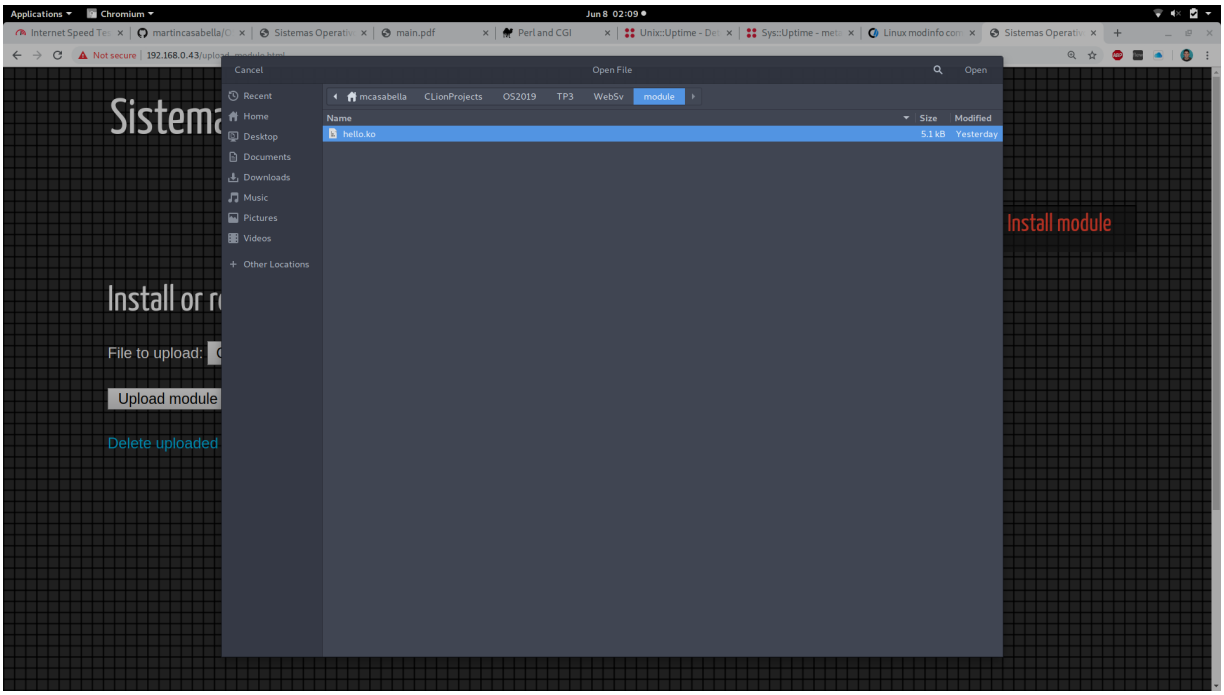

Figura 11: **Habiendo compilado el modulo, probamos enviarlo al servidor e instalarlo**

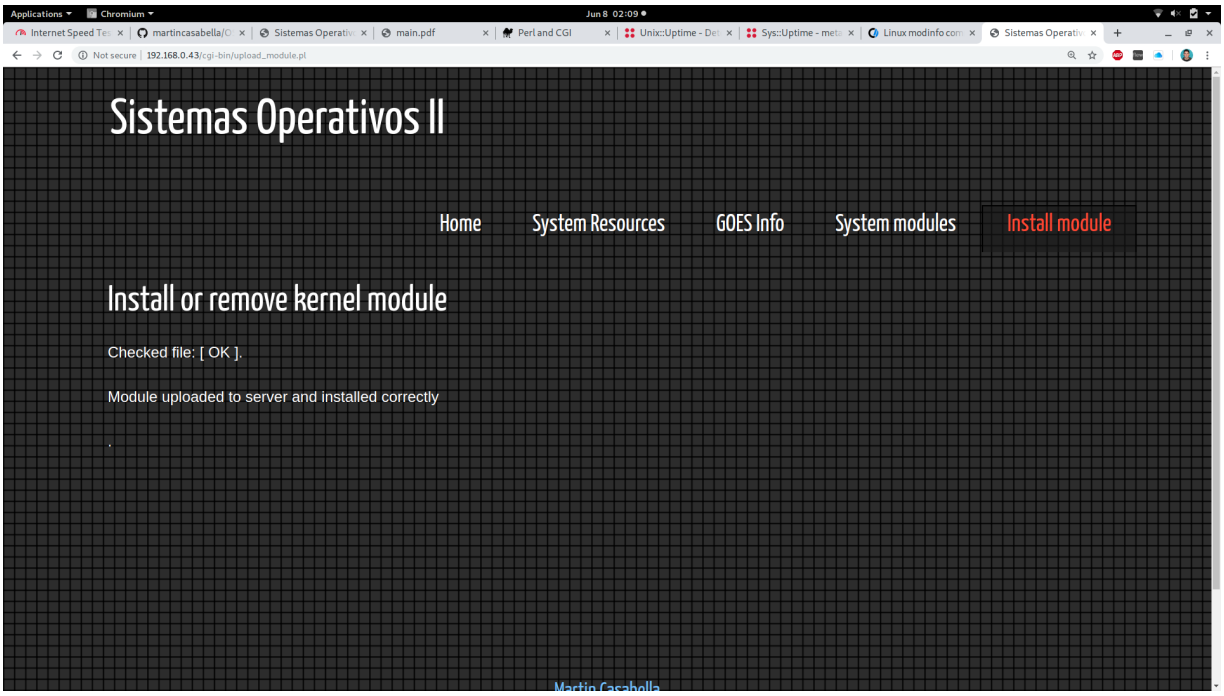

Figura 12: **Vemos como responde el server ante la carga**

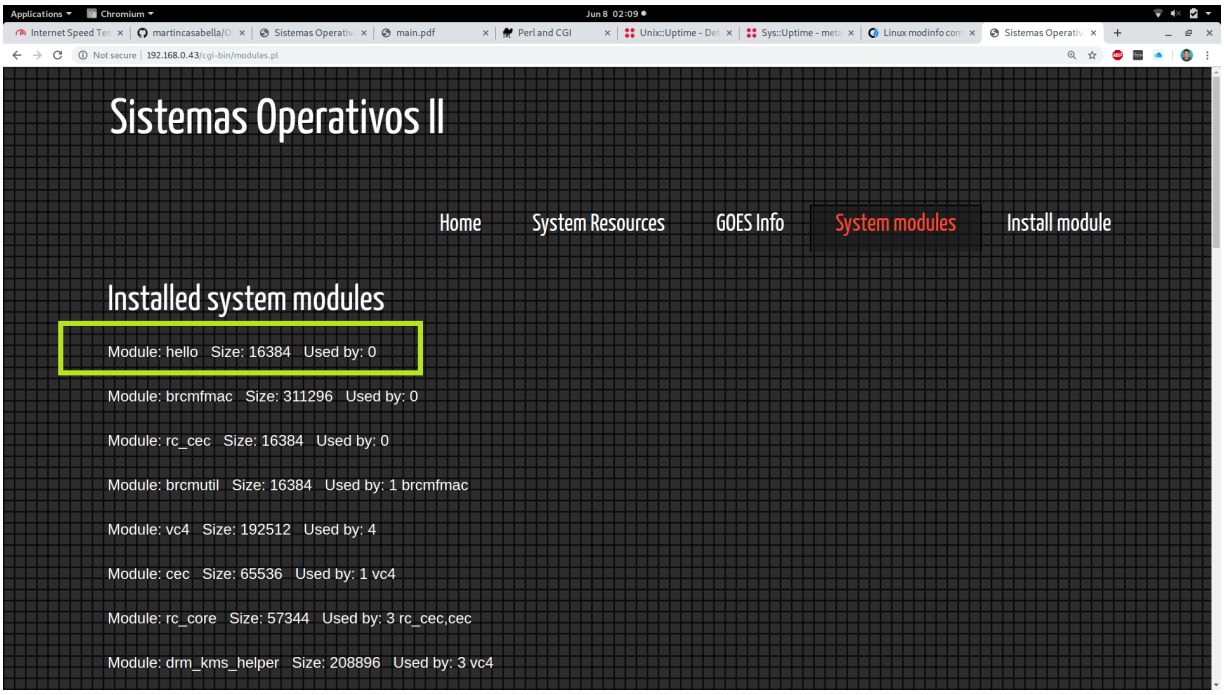

Figura 13: **Ahora vemos el nuevo modulo listado al volver a pedir la lista de modulos**

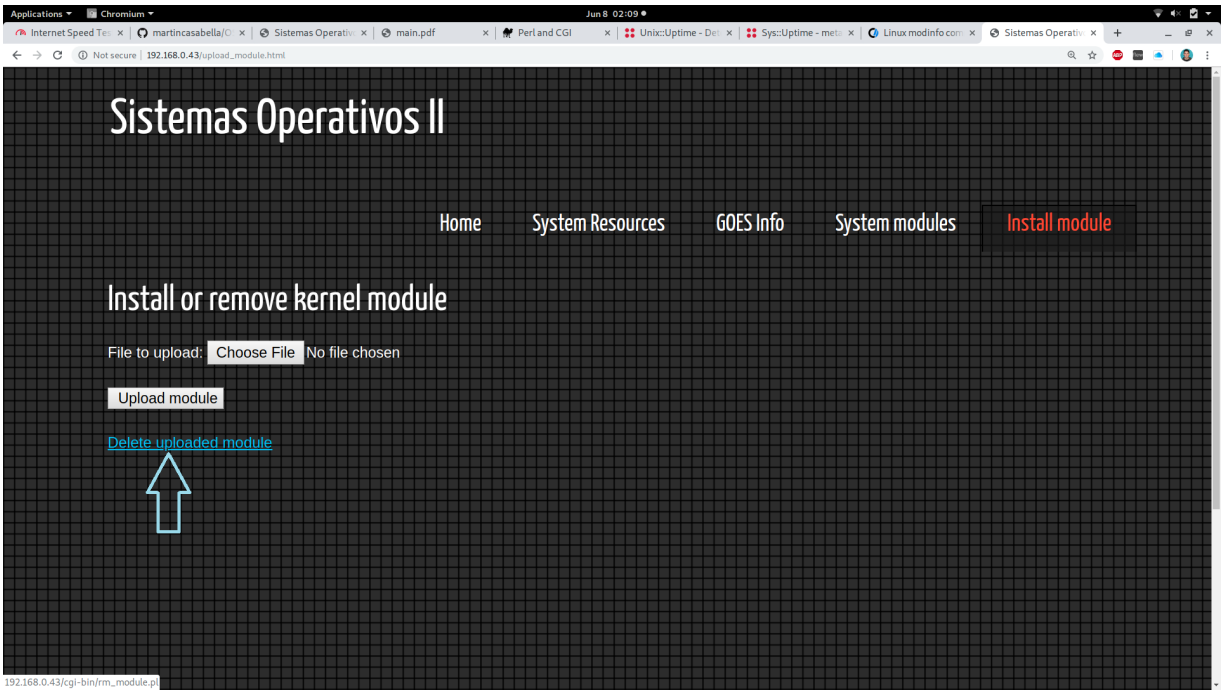

Figura 14: Procedemos a querer borrar el modulo

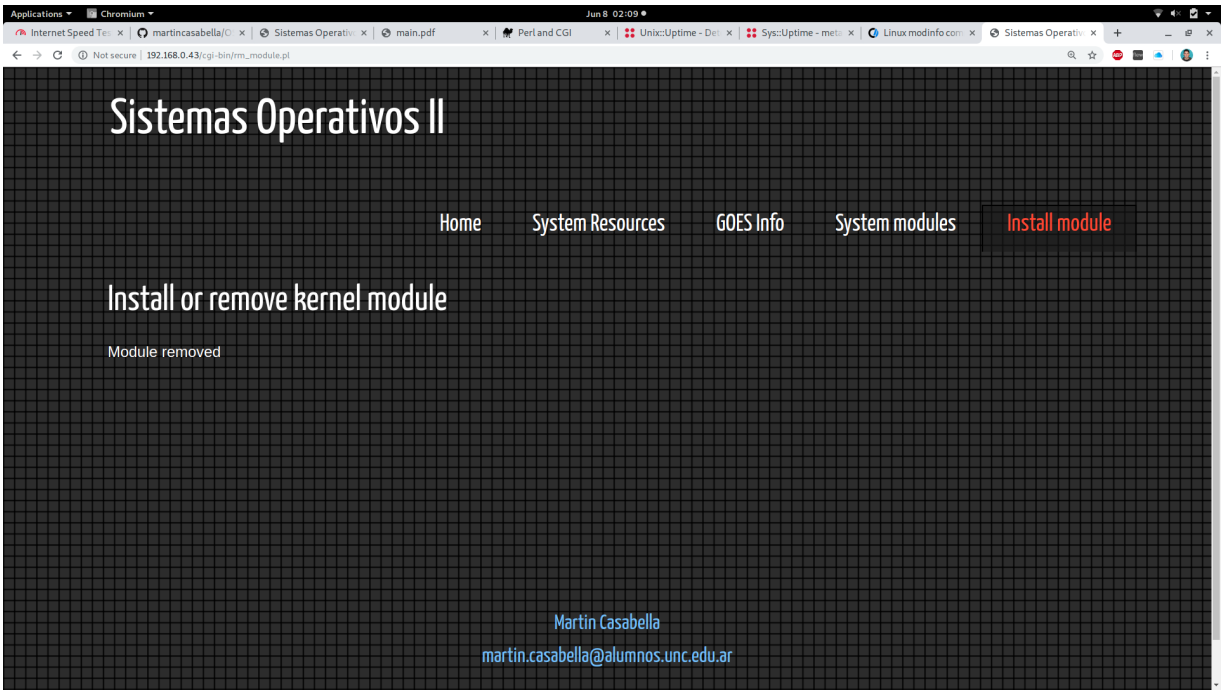

Figura 15: Obtenemos respuesta a dicha petición

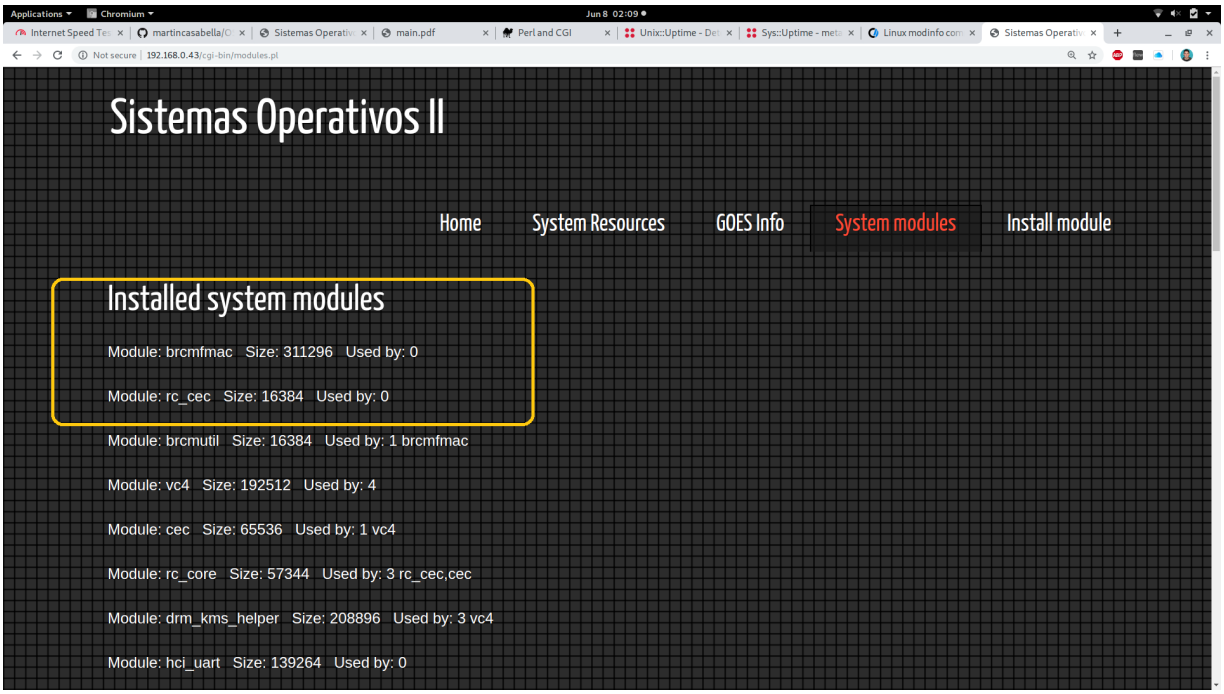

Figura 16: Vemos que el modulo fue removido exitosamente

Configuración de permisos

El servidor web debe ser capas de levantar y remover módulos del kernel, para esto hay que darle permisos para que utilice los comandos insmod, modprobe e insmod, aunque en verdad todos los comandos son los mismo ya que hace un enlace simbólico a Kmod.

Se procede a editar el archivo **sudoers** que contiene los permisos a comandos de los usuarios.

```
1 #ALIAS para los comandos
2 Cmnd_Alias MANAGEMOD = /usr/ bin / insmod , /usr/ bin / kmod , /usr/ bin / rmmod
3 # Al usuario http le permitimos desde todos los hosts (ALL), que ejecute comandos como
4 # root (root), sin solicitar password (NOPASSWD), los comandos definidos en el alias
5 # MANAGEMOD
6 http ALL =( root ) NOPASSWD : MANAGEMOD
```