Additional Files

apache2.conf

mime.types

Telematic Applications: Web Servers (HTTP)

In this assigment, we will configure an Apache Web Server version 2.4.X and test its functionality by sending HTTP requests and analyzing the responses. In the lab, we have an Apache server installed in the /usr/sbin/ directory. By default, Apache retrieves configuration information from the file `/etc/apache2/apache2.conf`. However, we will use our own configuration file to test different options and adapt it to our needs.

Above you have two files to download for the practice (`apache2.conf` and `mime.types`). The first one (`apache2.conf`) contains directives necessary to launch the server and specific directives related to its behavior. The second file (`mime.types`) contains the list of MIME types related to the content served by the server. The structure of this file is simple: type/subtype and the extension associated with that type.

Server Configuration

Before running the server, we need to modify several directives. For detailed information on any of them, you can refer to Apache Directives.

1. Prepare your account by downloading the necessary files into it and creating the following directory structure, where `d` indicates a directory or folder, and `f` indicates a file:

```
d-httpd                    (root directory of the server "
<tt>ServerRoot</tt>")
f--- apache2.conf
f--- mime.types
d--- defaultdocs
d--- log
d--- vhost1
d    --- docs
d    --- log
d--- vhost2
d    --- docs
d    --- log
```

-

Next, we will modify certain directives to point to files in our account and to determine certain execution parameters. Open `apache2.conf` and change the directives:

- **ServerRoot**: path to the directory where the necessary configuration files for the correct operation of the server are located (absolute path to the `httpd` directory in your account).

- **`ServerName`**: name of the machine where the server is hosted.
- **`Listen`**: specify as port 9xxx, where xxx corresponds to the last 3 numbers of the machine's IP address.

Other directives like **`DocumentRoot`**, **`LockFile`**, **`PidFile`**, **`ScoreBoardFile`**, **`ErrorLog`**, **`CustomLog`**, and **`TypesConfig`** indicate directory or file names (created by the server) relative to the path specified in **`ServerRoot`**.

-

Create a simple web page `aptel.html` in `httpd/defaultdocs`, for example:

```
<html>
<body>
¡Hola mundo!
</body>
</html>
```

-

Now we can **start our server** by specifying the adapted configuration file. Use the command:

```
/usr/sbin/apache2 -f $FULL_PATH/apache2.conf
```

/usr/sbin/apache2 -f /usr/lab/alum/0499081/httpd/apache2.conf

```
<p>
  Verify that the server is running: use the <tt>ps</tt> command or open the
<tt>apache2.pid</tt> file
  (if this file does not exist, the server is not running—check the logs to detect
the problem).
</p>
```

-

Access the newly created page (aptel.html) through the browser at the URL `http://servername:port/aptel.html`. Also, try accessing it using simply `http://servername:port/`.

```
<p>Note that if we do not specify the HTML page we want to access, the server can
act in two ways:</p>
<ul>
  <li>Showing a list of directories (not recommended), or</li>
  <li>Displaying a default HTML file.</li>
</ul>
```

```
<p>
  For the second behavior, the <b><tt>DirectoryIndex</tt></b> directive is used to
specify the name of the default file.
  Make sure this directive is already defined and is typically
<tt>index.html</tt>. For now, do not modify this directive.
</p>

<p><strong>Note:</strong> If there is an error, investigate what is happening by
accessing the file configured in the <tt>ErrorLog</tt> directive.</p>
```

HTTP Requests and Responses

Once we have verified the operation of our server using the browser as a client, let's use a `telnet` client.

6. Make the requests made earlier with the browser `http://your_machine:your_port/aptel.html` and `http://your_machine:your_port/` using `telnet`. Send requests using HTTP/0.9 and HTTP/1.0. What responses and headers do you get?

> >When we make a HTTP/0.9 request, we do not receive any header, only the "code" html code of the web.
> When doing the request using HTTP/1.0, we get the following headers: Date, Server, Last-Modified, ETag, Accept-Ranges, Content-Length, Connection, Content-Type.

- Repeat the requests using HTTP/1.1. What header should you include in the request? Does the server immediately close the connection? What should be included in the request for it to do so? Test it.

> > We must include the header HOST to make a HTTP/1.1 request, and also the server does not immediately close the connection, so we should close it previously on the request including the header Connection: close.

- Even if nothing is sent in the HTTP/1.1 request to immediately close the connection, the server does so after a period of inactivity. Measure how long this time is and check that it matches what is specified in the **`KeepAliveTimeout`** directive.

> > It corresponds with the value of the variable KeepAliveTimeout which has a value of 15 seconds, after that time, the connection is closed autommatically.

- Look at the `apache2.conf` file and explain the purpose of the **`KeepAlive`** and **`MaxKeepAliveRequests`** directives.

> > KeepAlive Off: The server closes the TCP connection immediatelly after sending a single file.
> KeepAlive On: The server keeps the TCP connection open after sending the file.
> MaxKeepAliveRequests (number): It defined the max. number of requests a client can make over a single persistent connection before the server forces it to close

Processes and Resources in the Web Server

10. Check how many processes are running. To do this, run the command `ps -x`.

```
<p>
  This number can be controlled with the <tt>StartServers</tt> directive. The
server initially starts this number of child processes to handle traffic,
  although it starts and stops new child processes dynamically as needed (the
maximum number is controlled by the <tt>MaxClients</tt> directive,
  also see <tt>MinSpareServers</tt> and <tt>MaxSpareServers</tt>).
</p>

<p>Modify the configuration file to start by default 8 child processes.</p>

<blockquote>
  <p>&gt; ps -x -o pid,ppid,tty,stat,time,command</p>
  <p>&gt; The paramter StartServers was initially set to 5.</p>
</blockquote>

<p>
  <strong>Note:</strong> Remember that changes to the server will take effect once
it is restarted. To stop the server, use the <tt>-k stop</tt>
  option of <tt>/usr/sbin/apache2...</tt> or the <tt>kill</tt> command along with
the process number.
</p>
```

Logs Management

Apache server generates interesting logs about its operation. On one hand, the error log and on the other, the access log.

11. The error log is in the file indicated by the **`ErrorLog`** directive. The **`LogLevel`** directive can vary the level of detail in the log. Examine the error log file and increase the level of detail to "debug", restart the server, and check how, for the same type of requests, a greater number of records is stored.

    > > We have changed LogLevel from warn to debug to get a more accurate error log.

- The **`CustomLog`** directive indicates the file where access to our web will be recorded. Examine the access log file. As you can see, machines are identified only by their IP. Find out which `apache2.conf` directive you need to change to ensure that the access log displays the machine's name ("hostname") instead of the IP. Make the change and verify that it works.

> > We have changed **HostnameLookups** from Off to On in order to substitute the IP address of the server with its name in the access.log file.

Content Types in Apache

As you may have seen when making the queries from questions 6 and 7, the server tells us the MIME type (Content-Type) of the object it sends. To know the MIME type of an object, check its extension in the file indicated in the **`TypesConfig`** directive.

13. Create a copy of the `aptel.html` file with the name `test.aptel` and change the directive so that `test.aptel` is the default file sent. Make requests to `http://your_machine:your_port/test.aptel` with a browser and with telnet. How do you see the file in the browser? What MIME type is the file sent with? Why?

    > > The file is displayed in the same way as before in the browser.
    > > We cannot see its MIME type as we have not set it as aptel on mime.types.

- Change the configuration so that when the extension is "aptel," it returns the MIME type `application/x-type-aptel`. Check that it works correctly. What does the web browser do now? Test with different browsers, e.g., Chrome and Firefox.

> > After defining the type in **mime.types** (application/x-type-aptel aptel), we can see in the header Content-Type the MIME type.

Directory Management and Security

The configuration for a specific directory can be changed using the `<Directory>` directive. For example, add the following code at the end of the apache2.conf file (replacing `DOCUMENT_ROOT` with the full path to the appropriate directory according to your web server configuration):

```
<Directory DOCUMENT_ROOT/internal>
    AuthType Basic
    AuthName "Telematic Applications"
    AuthUserFile SERVER_ROOT/passwd
    Require user aptel
</Directory>
```

15. Create the `DOCUMENT_ROOT/internal` directory (within your folder specified in the `DocumentRoot` directive, `defaultdocs`) and copy the `aptel.html` file into it. Remember to add this filename to the default files that are sent. What happens now when you try to access the URL `http://your_machine:your_port/internal` with a browser?

    Note: You can add a small text (for example, change the message to ¡Hola Mundo interno!) to the file if you want to differentiate it.

    > > We cannot access the request, as we are asked for a user and a password.

- You will need to generate the password file (`SERVER_ROOT/passwd`), which can be done with the `htpasswd` utility. Use this utility to create a file with a user `aptel` and password `redes`. Try to log in with this user and password using a web browser.

Run this command at folder httpd

> > htpasswd -c -b passwd aptel redes

17. Now try it using telnet. What is the realm name? Where does the server get it from? Send a second request to access the page. You can use this Base64 converter.

aptel:redes → YXB0ZWw6cmVkZXM=

```
telnet localhost 9215

GET /internal HTTP/1.1
Host: your_machine
Authorization: Basic YXB0ZWw6cmVkZXM=
```

HTTP/2

18. Make an HTTP/2 request to the website `https://example.com` using `curl` and `nghttp`. Ensure the request is made using HTTP/2.

   - `curl -v --http2 https://example.com`

- What headers are sent in the request?

Pseudo-headers:

> >:method: GET
> :path: /
> :scheme: https
> :authority: example.com

Regular headers

> >user-agent: curl/7.88.1
> accept: */*

- What version of TLS does the website use?

> >TLSv1.3

- How many streams are there?

> >only 1 (expected for a single-page request with no server push)

- What frame types are exchanged?
- What differences can be observed between using the `curl` command and `nghttp`?

Redirection and Virtual Hosts

19. The **`Redirect`** directive allows redirecting a client from one URL to another. Use this directive to make browsers visiting the URL `http://your_machine:your_port/old/` automatically redirected to `http://your_machine:your_port/new/`.

ADD:

> > **Redirect** permanent /old/ http://localhost:9215/new/

to apache2.conf

-

For this part, it is required to modify the directives **within the `VirtualHost` directive**: `DocumentRoot`, `ServerAdmin`, and `ServerName`.

```
<p>
  The <b>&lt;VirtualHost&gt;</b> directive allows the server to respond to
different web pages, depending on the server name used to access it. This name is
sent with the Host header of HTTP/1.1.
</p>

<p><strong>ANSWER:</strong></p>

<p><strong>Goal:</strong> Configure two name-based VirtualHosts and test them with
telnet (HTTP/1.1 Host header).</p>

<ol>
  <li>
    <p><strong>1) Create directories</strong></p>
    <pre><code>mkdir -p /usr/lab/alum/0499081/httpd/vhost1
```

mkdir -p /usr/lab/alum/0499081/httpd/vhost2

```
  <li>
    <p><strong>2) Create pages</strong></p>
    <pre><code>echo "¡Hola mundo vhost1!" &gt;
/usr/lab/alum/0499081/httpd/vhost1/aptel.html
```

echo "¡Hola mundo vhost2!" > /usr/lab/alum/0499081/httpd/vhost2/aptel.html

```
  <li>
    <p><strong>3) Add VirtualHosts</strong> (append at the end of
<code>/usr/lab/alum/0499081/httpd/apache2.conf</code>)</p>
    <pre><code>&lt;VirtualHost *:your_port&gt;
ServerAdmin webmaster@vhost1.local
ServerName vhost1.local
DocumentRoot /usr/lab/alum/0499081/httpd/vhost1
```

</VirtualHost>

<VirtualHost *:your_port> ServerAdmin webmaster@vhost2.local ServerName vhost2.local DocumentRoot /usr/lab/alum/0499081/httpd/vhost2 </VirtualHost>

```
  <li>
    <p><strong>4) Reload Apache</strong></p>
    <pre><code>sudo systemctl reload apache2</code></pre>
  </li>

  <li>
    <p><strong>5) Test vhost1 with telnet</strong></p>
    <pre><code>telnet your_machine your_port</code></pre>
    <pre><code>GET /aptel.html HTTP/1.1
```

Host: vhost1.local

Expected body: `¡Hola mundo vhost1!`

```

    <li>
      <p><strong>6) Test vhost2 with telnet</strong></p>
      <pre><code>telnet your_machine your_port</code></pre>
      <pre><code>GET /aptel.html HTTP/1.1
```

Host: vhost2.local

Expected body: `¡Hola mundo vhost2!`

```

    <li>
      <p><strong>7) (Optional) Check redirect/default behavior</strong> (wrong/empty
   Host)</p>
      <pre><code>telnet your_machine your_port</code></pre>
      <pre><code>GET /aptel.html HTTP/1.1
```

Host: unknown.local

Note: Apache usually serves the *first* VirtualHost as default.