World Wide Web Technology Fundamentals

by Ted Kosan

Table of Contents

1 A web on top of a net

1.1 As we learned in the “Internet Technology Fundamentals” book, the Internet consists of millions of computers that are able to send messages to each other through various communications mediums. Each computer has an **IP address** (so it can be located on the network) along with a group of **ports** that sit between the **process** that are running on the computer and the **network**. Some of these processes are designed to provide **services** over the network, and they are usually bound to **well known** port numbers or **registered** port numbers. Messages that contain requests for a given **well known** or **registered** service include the port number for that service so that they can be routed to the service's process.

1.2 There are currently thousands of services that are included in IANA's **well-know** and **registered** ports list (http://www.iana.org/assignments/port-numbers) but the service which is associated with **port 80** has become especially important since its creation in the early 1990s. The service is called **HTTP** (Hyper Text Transfer Protocol), and it implements a protocol for copying files from one machine to another on the Internet. Most people, however, know this service by its more popular name which is a **web server**.

1.3 Web servers are designed to listen on a port (usually port 80), accept requests for files that they hold, and then send copies of the requested files to the client/requester using the HTTP protocol. The most popular client application is a **web browser**. Examples of popular web browsers are FireFox and Internet Explorer.

1.4 The **World Wide Web** (WWW) consists of the millions of servers in the world that have an **HTTP** service bound to **port 80**. The **HTTP protocol** rides on top of the **TCP/IP protocol**, and this is how the Web can be thought of a being on top of the Net.

2 The Apache HTTP server

2.1 Note: Make sure you have done all the steps in section 42 (Configuring The Network) of the “Installing Gentoo Linux x86” v2.01 book before proceeding.

2.2 Numerous server applications have been written that implement the HTTP protocol and can provide it as a service through a port (usually port 80). These applications are know as **HTTP servers**. They are also known as **web servers.** The most widely used HTTP server, however, is the open source

Apache server, and its popularity can be seen in the charts at this website (http://news.netcraft.com/archives/web_server_survey.html). Since Apache is currently the most widely used HTTP server, this is the server that is used in this document.

2.3 You will now install the Apache HTTP server on your Gentoo system, and the first step in this process is to switch to the superuser account:

```
tkosan@kosan1 ~ $ su
Password:
kosan1 / #
```

2.4 If are using the Atlas network in the Advanced Technology Center, and your VirtualBox network is set to “bridged”, then you will need to use the **lynx** text-based browser to authenticate with the Atlas network.

```
kosan1 / # lynx
```

2.5 Read the instructions at the bottom of the lynx application to learn how to use lynx. Then use lynx to authenticate with the Atlas network.

2.6 Install Apache on your system using the **emerge** command:

```
kosan1 / # emerge apache
```

2.7 After the emerge process is finished, change into the **/etc/init.d** directory and notice that a script for controlling the server called **apache2** has been placed there:

```
kosan1 / # cd /etc/init.d

kosan1 init.d # pwd
/etc/init.d

kosan1 init.d # ls
apache2            fsck               loopback    pydoc-2.7    sysctl
bootmisc           functions.sh       modules     pydoc-3.3    sysfs
busybox-ntpd       hostname           mount-ro    reboot.sh    syslog-ng
busybox-watchdog   hwclock            mtab        root         termencoding
consolefont        ip6tables          net.enp0s3  rsyncd       tmpfiles.dev
crypto-loop        iptables           net.eth0    savecache    tmpfiles.setup
devfs              keymaps            net.lo      shutdown.sh  udev
dhcpd              killprocs          netmount    sshd         udev-mount
dhcrelay           kmod-static-nodes  numlock     swap         urandom
dhcrelay6          local              pciparm     swapfiles    vixie-cron
dmesg              localmount         procfs      swclock
```

2.8 Lets start the apache server and then use a web browser to connect to the server to make sure it is running. **Start** Apache using the following

command:

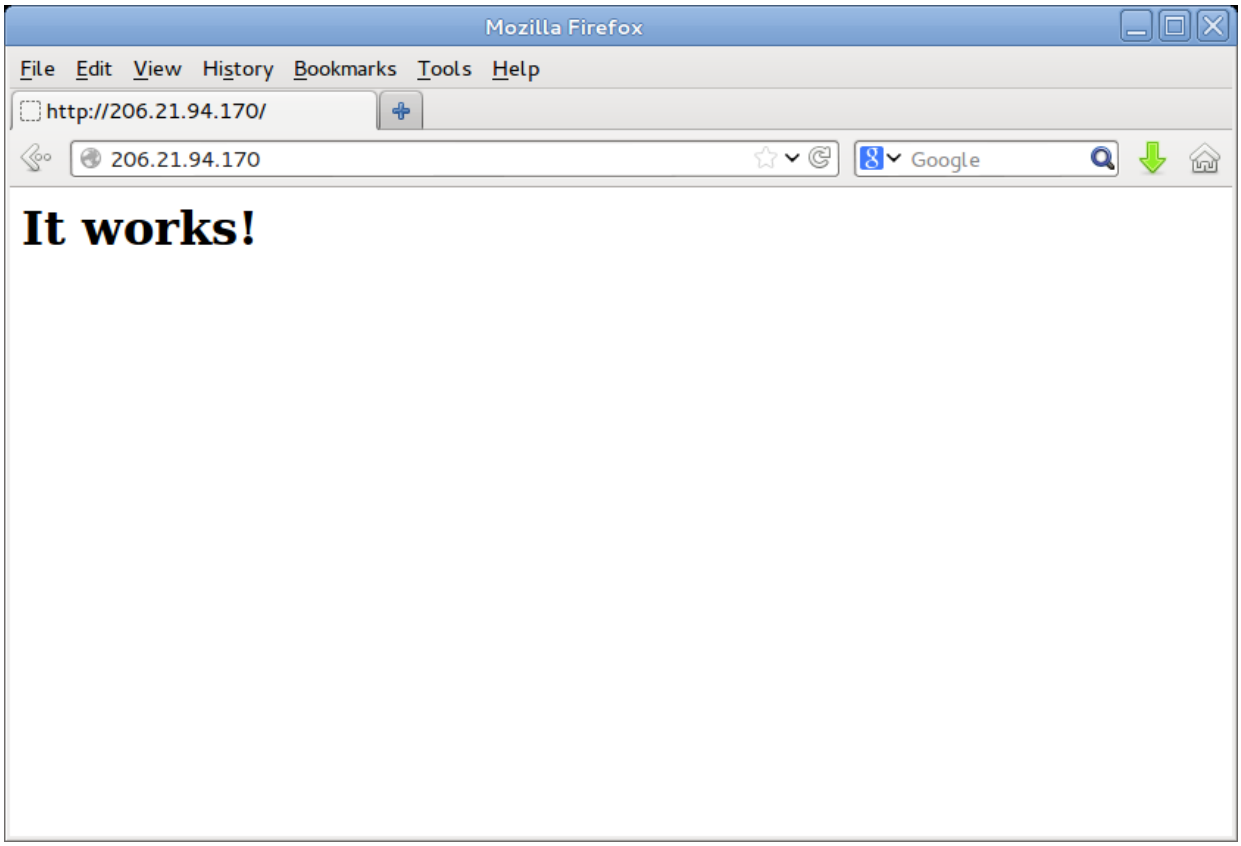
```
kosan1 / # /etc/init.d/apache2 start
 * Starting apache2 ...                                                [ ok ]
```

2.9 Now that the server has been started, you will need to determine the IP address of your machine so that you can type it in your browser's URL bar:

```
kosan1 / # ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 206.21.94.170  netmask 255.255.255.0  broadcast 206.21.94.255
        inet6 fe80::a00:27ff:fe28:853a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:28:85:3a  txqueuelen 1000  (Ethernet)
        RX packets 128465  bytes 29289032 (27.9 MiB)
        RX errors 0  dropped 99  overruns 0  frame 0
        TX packets 4226  bytes 318322 (310.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 16436
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 6  bytes 328 (328.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 328 (328.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2.10 The IP address of **my** machine is **206.21.94.170** so I would type **http://206.21.94.170** into my browser's URL bar (notice that **http://** is telling the browser that we want it to access the server using the **HTTP protocol**). **Note: if your system has an IP address that has a 10 as its first number (for example 10.0.1.5), you will need to halt your system, switch your virtual machine's network to "bridged", and then reboot.** Here is the default web page that the Apache server will send to your browser when you contact it:



3 Creating a small HTML file and placing it in the Apache server's webroot directory

3.1 The Apache server's **webroot** directory is the subdirectory in the computer's directory hierarchy that the server serves files from. When Apache is first emerged, its **webroot** directory is set to **/var/www/localhost/htdocs**. Change into the **/var/www/localhost/htdocs** directory and lets see what it contains:

```
kosan1 / # cd /var/www/localhost/htdocs

kosan1 htdocs # pwd
/var/www/localhost/htdocs

kosan1 htdocs # ls
index.html
```

3.2 If a directory that an HTTP server is serving files from contains a file named **index.html**, this is the default file that is served from the directory if no specific file is requested. Let's delete all of the files in this directory and then create and place our own file here. Execute a **rm index.html** command

in order to delete the index.html file in the current directory (make sure you are in the /var/www/localhost/htdocs directory before doing this):

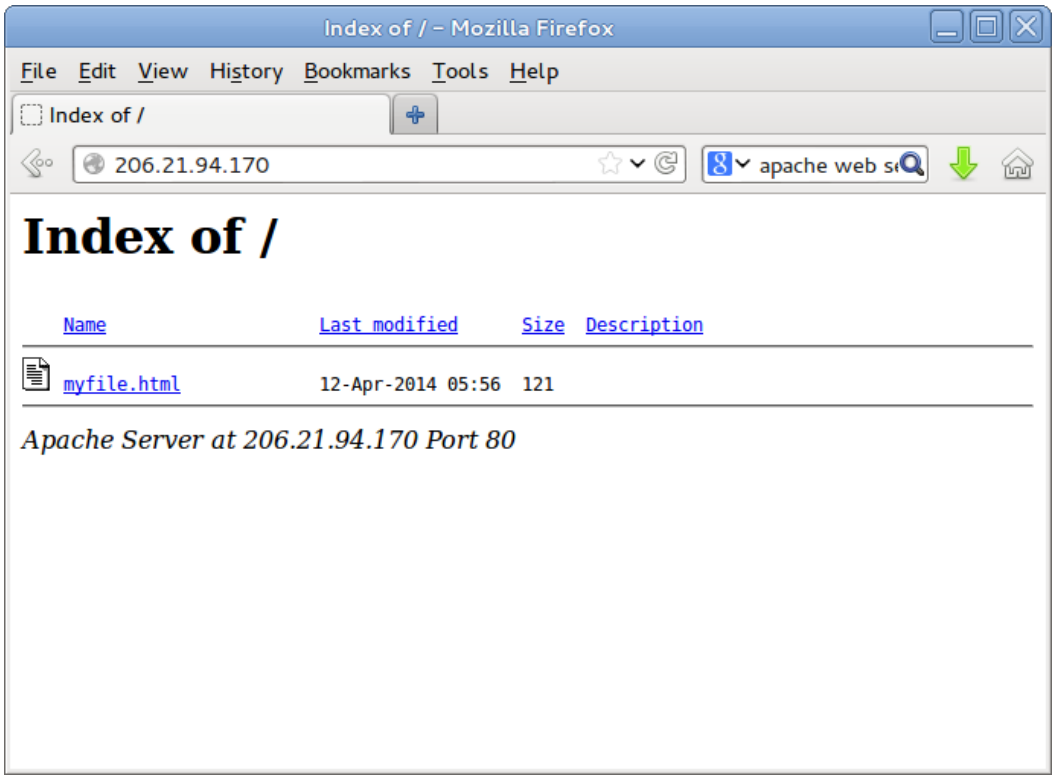
```
kosan1 htdocs # rm index.html

kosan1 htdocs # ls
<The directory is now empty>
```

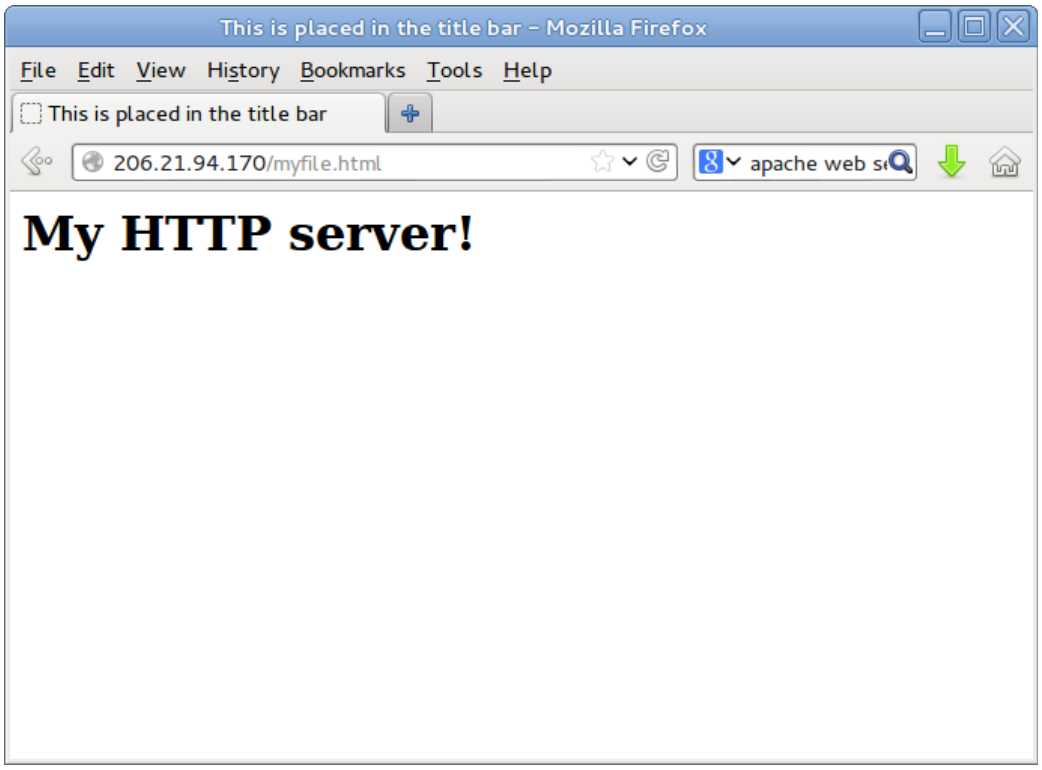
3.3 Use **nano** to create a file in the **/var/www/localhost/htdocs** directory named **myfile.html** which contains the following text:

```
<html>
  <head>
    <title> This is placed in the title bar </title>
  </head>
  <body>
    <h1>My HTTP server!</h1>
  </body>
</html>
```

3.4 After you save this file, if you go back to the browser and hit the **refresh** button, the following page should now be shown:



3.5 This is what Apache will serve when no specific file has been requested from a directory, and the directory does **not** contain an **index.html file**. You can click on the **myfile.html** link in the browser to open the page you just created, or you can enter **myfile.html** after the IP address in the URL bar:



3.6 The forward slash '/' after the IP address represents the top-level **webroot** directory of the server (which is /var/www/localhost/htdocs). If you create subdirectories inside the **htdocs** directory (using the **mkdir** command), you can add their paths after the '/' symbol in order to access them. For example, if you created a subdirectory inside of **htdocs** called **pictures**, then you could type something similar to **http://206.21.94.170/pictures** in order to access its contents.

3.7 The text that was placed into the **myfile.html** file is in a format called **Hyper Text Markup Language** (HTML), and it consists of tags that indicate how the text they contain should be rendered by a browser. For example, the **<h1><h1/>** tag in this file has a browser render the contents of this tag as a bold text heading in a large font. HTML is beyond the scope of this document but there are numerous tutorials on the Internet that contain information about it. Feel free to research HTML on your own and then create new HTML files inside of the **htdocs** directory so that Apache can serve them.