LESSON E24_EN. HOW TO CREATE A WEBSITE.

Parent Entity: Consorzio Pisa Ricerche – Divisione Informatica e Telecomunicazioni
Via Turati 43/45 56125 PISA (ITALY)
Tel. +39 050 915811
Fax +39 050 915823
Authors: Cristiano Bozzi, Gino Carrozzo, Nicola Ciulli, Giodi Giorgi, Gianluca Insolvibile, Alessandro Martucci, Giacomo Sergio
Consultations: Every working day between 9.00 to 12.00 a.m

Studying this lesson you will learn:

- How to create a simple web site and how to make it work, from necessary services installation and configuration to deployment process of a site that must be accessible by the World Wide Web.

CONTENT OF THE LESSON

1. How to configure DNS.
2. How to create web site examples for testing.
3. Dynamic content
4. How to let the owner of the hosted website administrate web facilities.

LEARNING OBJECTIVES:

After learning this lesson you will be able to:

- manage DNS related issues,
- basically configure the server,
- develop of an example site,
- make the site work.

1. HOW TO CONFIGURE DNS

1.1. Introduction

The Domain Name System (DNS) is a distributed database. Particular applications, called Name Servers, are the server part of the client-server architecture on which DNS is based. Name Servers contain certain database data made available to the clients, called “resolvers”. These ones are simple applications that make and send requests to name servers.

The structure of a DNS database is very similar to the one of a filesystem [Figure 1.1]

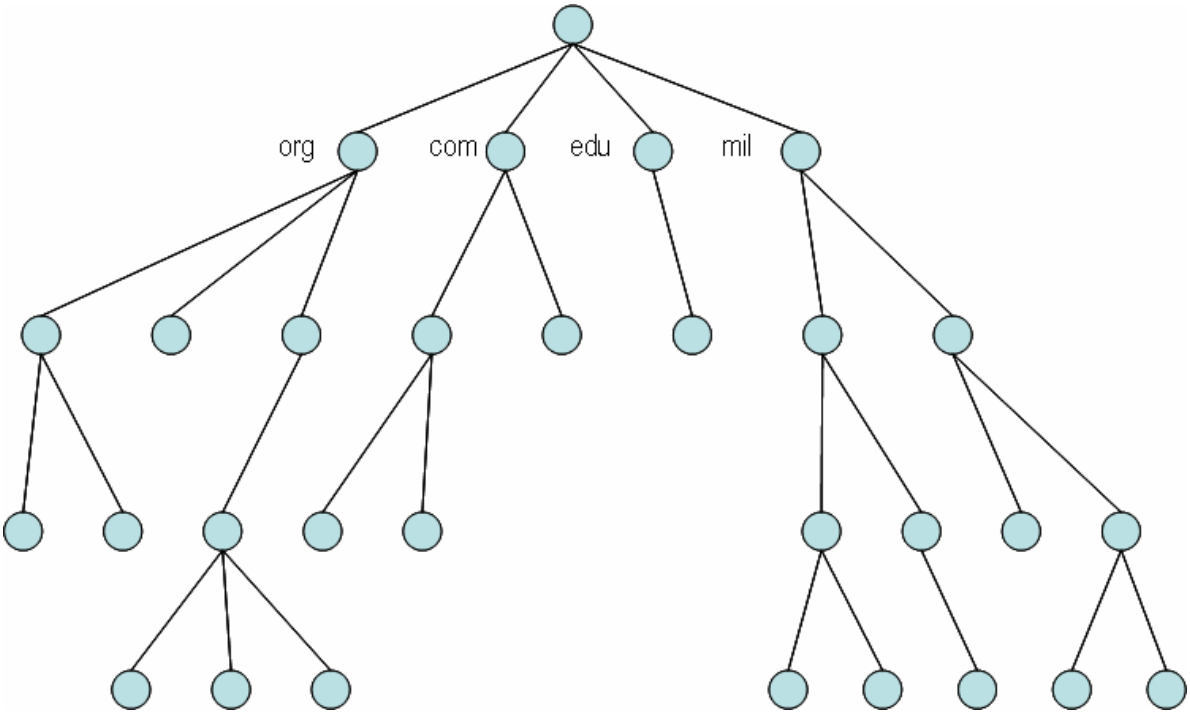

Figure 1.1 DNS structure

The entire system can be represented by an upside down tree with the root in its higher part.

Each node has a label that determines the node relatively to its father node. Each sub-tree represents a portion of the whole database, that is a "domain" inside the Domain Name System. Each domain, as each sub-tree, can be divided again into sub-domains.

Each domain owns a univocal name, which locates its position inside the distributed database. So, inside the DNS, the domain name corresponds to a sequence of labels that go from the root of the specific sub-tree to the root of the entire tree, separated by '.' characters.

Inside the DNS each domain can be divided into sub-domains: their management can be the task of different organizations. For example, the organization that manages "edu" domains, called EDUCASE, delegates its sub-domain management to Berkeley University ("berkeley.edu") [Figure 9].

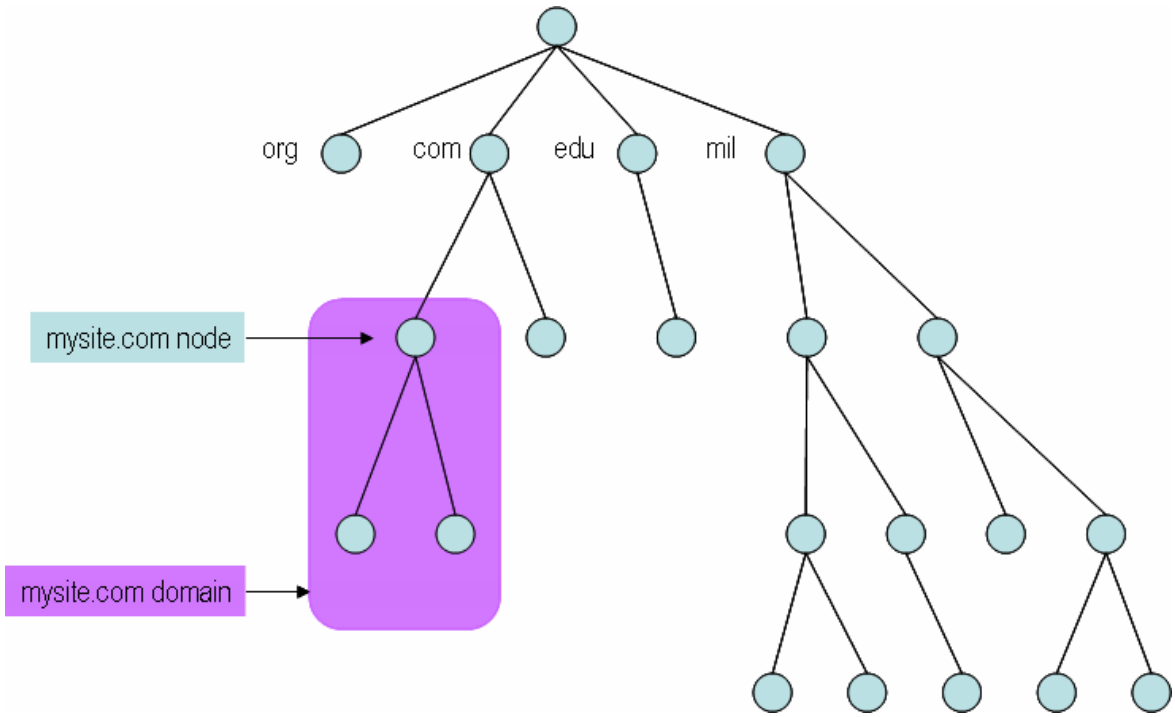

Figure 1.2 creation of a new zone

This authority delegation practically creates a new "zone", that is a part of the names space autonomously administered.
The new zone will include all domain names ending with "berkeley.edu". Moreover this domain can be divided into sub-domains, i.e. "cs.berkeley.edu", and some of them can be configured as separated zones. In other words, if "cs.berkeley.edu" is an independent zone, then "Berkeley.edu" zone will not include domain names ending with "cs.berkeley.edu".

It is worth noticing that this structure brings to considerable advantages. One of the most relevant is the impossibility of having name collisions; each domain has a univocal name in the general structure. The organization managing the domain is free of assigning names to hosts and to sub domains inside its domain. Any chosen name will not be in conflict with domain names of other organizations because it will end with the univocal organization name.

Domains, besides being considered in terms of domains and sub domains, can be also considered as "levels". The term only refers to their position inside the main domain:

- A "*top level"* domain, indicated as "*top-level domain*", is a son node of the root node;
- A "*first-level*" domain, is a son of the "*top-level domain*";
- A "*second-level*" domain, is son of a "*first-level domain*", and so on.

Considering that the DNS is essentially a distributed database, domain names can therefore be considered as "indexes" inside the DNS database. It is also possible to think of DNS information as associated to a domain name.

Each host on the Internet has a domain name that points to host data (i.e. IP address). Such data are included in, so-called, "resource records" (RRs).

There are situations, or better network configurations, in which introducing a DNS is just a useless complexity introduction. If, as in our case, we need Internet connections, the DNS is essential.

It is possible to think of the DNS as a sort of universal language of the Internet. Almost any service of the Internet uses the DNS (web, email, ftp, etc.).

It is worth noticing that the types of information that can be collected using a domain name tightly depend on the name context. For example, sending a mail to someone c.o. “google.com” will tell us something about the mail routine, while sending a “ping” packet to the domain name will give us data regarding the specific host (IP address).

1.1.1. Domain name registration

In our context the domain name is an unambiguous address that locates the web site on the Internet.
For example, the web address or the URL of our site might be:

http://www.mysite.com/

In this case “mysite.com” is the domain name. Domain names are registered (by owners) to agencies recognized to perform this kind of operations.

Let us now try to understand why domain names are used for web sites.

There are millions of web sites and each of them resides in a web server. Each web server can be identified by an univocal IP address.
Detecting a server for a specific web site requires remembering the IP address of such server: addresses, as known, are made of numbers and so are difficult to be remembered.
Therefore in 1985 domain names have been introduced to create an easy to remember relationship between server address and server location.

Each domain name is registered through an agency by ICANN (http://www.icann.org)

ICANN (Internet Corporation for Assigned Names and Numbers) is a no-profit international agency whose main tasks consist on assigning IP (Internet Protocol) addresses and managing Top-Level and generic level (gTLD) domain name system and international codes (ccTLD) system.

This agency keeps a registered domain names database and name server locations of those sites. This information is stored into the DNS (Domain Name System). The domain name and related IP address information is stored in a distributed manner on many servers so to be easily consulted.

The list of agencies delegated to domain registration can be consulted at “http://www.internic.net/regist.html”.

When an Internet user types the URL of a site, as the one mentioned in the example, the web browser (or web client) sends a request to the nearest DNS server. Such request is used to obtain the IP address corresponding to the indicated domain name. If the DNS does not manage to resolve the domain name location, sends the request to the hierarchically superior DNS.
The process goes on until the server hosting that web site is found.

If the server is not found, the DNS server returns an error message or redirects the user to another location. The browser can then interact with the web server using the IP address and requesting specific web pages. Afterwards such pages are downloaded on the web browser and displayed on the user computer.

The domain name is made of two parts: the first level domain, usually called “top-level-domain” (TLD) and second level domain. The TLD is typically indicated as the domain extension (i.e. “com”, “net”, “org”). In the example above considered, “mysite” is the second level domain and “com” is the extension, or TLD. The two parts are separated by a “.” character.

Considering the context of this lesson, it is worth noticing that in many cases the domain name of a web site is very “communicative” on the Internet. Therefore it is necessary that the web site owner chooses an appropriate domain name and extension.

1.1.2. DNS software

The next step consists on (if own zones configuration and name server managements are scheduled) choosing the software that performs the function of name server.

Microsoft has its own name server, Microsoft DNS server, included in Windows operative system. Therefore we can find several versions of BIND (Berkley Internet Name Domain), a DNS server typically used in Unix systems. BIND is a software system that implements both client and server parts.

The client part is called “resolver” and creates domain name requests to be sent to the server. The software server part replies the requests made by the “resolver” and is implemented by a daemon called “named”.

nslookup

Therefore there is a debug tool, provided as part of the BIND package, called “nslookup”.

”The **nslookup** command can be used in Windows and Unix to find the IP address of a particular computer, using lookup” http://en.wikipedia.org/wiki/NSLOOKUP

It allows directly consulting the name server and collecting data known by the DNS system.
It is a tool used to determine wether the server is correctly working or nor or to collect data from remote servers.

The “nslookup” program is used to resolve interrogations both in an interactive manner and directly from command line. In the following figure [Figure 10] an example is shown. It shows how to obtain the IP address of a host using the command line.

```
gino@fry:~$
gino@fry:~$
gino@fry:~$
gino@fry:~$
gino@fry:~$
gino@fry:~$ nslookup www.google.com

Server:         131.114.33.148
Address:        131.114.33.148#53

Non-authoritative answer:
www.google.com  canonical name = www.l.google.com.
Name:   www.l.google.com
Address: 66.102.9.99
Name:   www.l.google.com
Address: 66.102.9.104

gino@fry:~$
```

Figure 1.3 *nslookup* utilization example

In this example a user tries to achieve the address of a machine through its name using “nslookup” program. “nslookup” displays name and address of the server used to resolve the interrogation and then returns the answer to the interrogation.

In the above example the system prompt is gino@free:~
And the command is the following:
gino@fry:~ **nslookup** **www.google.com** (Enter)

The “nslookup” command is often used in an interactive manner, which shows its real usefulness.

In fact in such modality it is possible to collect many data, whose detail will be shown in the following, after their description.

Wishing to enter interactive mode, it is just required to type *nslookup* in the command line. Otherwise it is possible to quit the session by ‘CTRL-D’ or typing *exit.*

In [Figure 1.4] the previous example is shown in interactive mode:

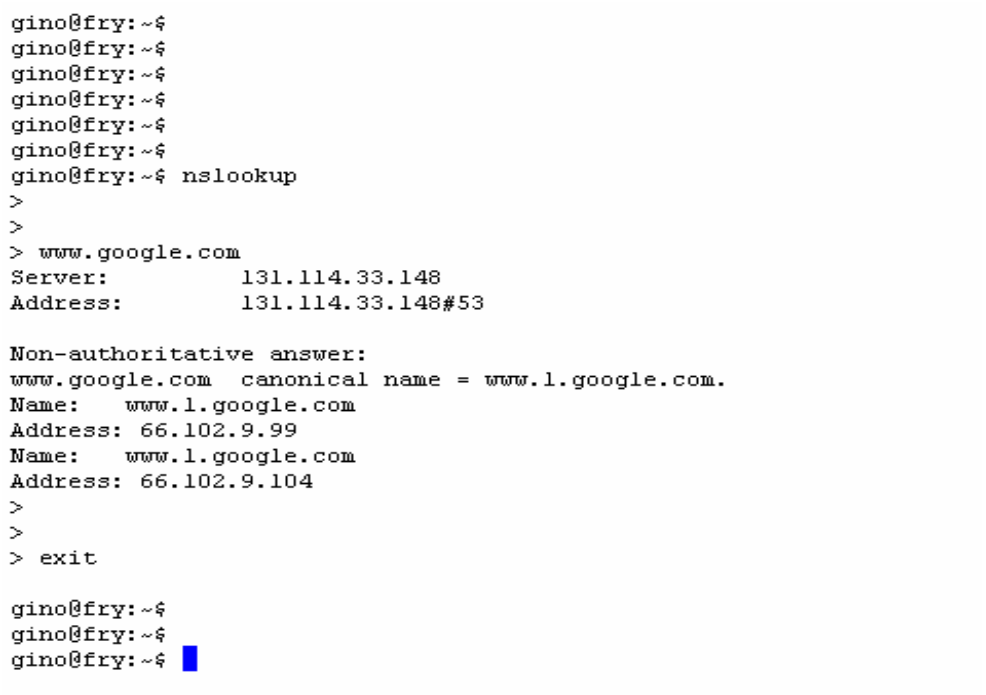

Figure 1.4 example of interactive modality use

In the above example, the nslookup offer the new prompt >tasting Enter, after the system response, lead at the apparition of the prompt consisting of the following character
With the command **exit** (Enter), the system return at the prompt gino@fry:~$

For the using nslookup in MS ® Windows please look at
http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/nslookup.mspx?mfr=true
whois

Another program that allows making DNS interrogations is “whois”.

It is used to collect data stored in the registering agency database.

”WHOIS is a TCP-based query/response protocol which is widely used for querying a in order to determine the owner of a domain name, an IP address, or an autonomous system number on the Internet.” http://en.wikipedia.org/wiki/Whois

Among these data there are some related to the domain owner, name servers etc.

Due to ICANN, such data should be public. On the Internet there are many sites that make this type of interrogations returning the result through a HTML page [Figure 1.5].

Anyway, because of high spam risks, some agencies let their registered users hide their data.

Figure 1.5 “whois” result example

1.2. DNS setup

The domain name configuration basically requires three operations to be executed:

- Register the domain name through a registration agency;
- Insert the domain name inside the DNS server as a db entry;
- Configure the web server to be able to accept requests for that domain name.

A domain request, for example http://www.mysite.com/, starts from one of registration agency servers, which forward the request to a DNS server containing data of that domain.

Once the DNS record (related to that particular domain) has resolved the domain name into a particular IP address, the request is sent to the web server listening on that IP address. Now the web server can forward the request received to the particular web site, basing on the domain name included into the request header.

As previously mentioned, a domain requests to be registered by an agency recognized by ICANN (http://www.icann.org) paying an annual fee. There are registration agencies for each specific national TLD (for example ‘co.uk’ for the United Kingdom). During the registration process, usually made online, the following data must be provided:

- Domain name owner data (name, company, address, phone number, email address, ...);
- Data of the administrative representative;
- Data of the technical representative;
- Details regarding the name server.

The DNS server, usually called “name server”, is given by the company providing the web hosting.

The DNS server must also have entries regarding the domain name registered in accord to the following.

At least two name servers which correspond, each, at the own IP Adress are required for a correct system functioning: for example “nameserver1.mysite.com” and “nameserver2.mysite.com”.

The domain setup on the DNS server can be executed after domain registration. However the domain name will not be active until the DNS setup is not completed. The DNS server relates a domain name to the IP address (or IP addresses) of the web server (or servers), so that a client can connect to a server using the domain name and not the IP address.

1.2.1. DNS software configuration *look also to Domain Name System*

http://en.wikipedia.org/wiki/Domain_Name_System

The basic idea on which domain name setup is based consists on creating a "zone" in which resolution search is done.

The following entries must be added inside such "zone":

- Start of Authority (SOA);
- Name Server (NS);
- Hostname (A);
- Canonical Names (CNAME);
- Mail eXchange (MX).

It is important to notice that information about the created zone is saved to a "zone file", also called "zone datafile".

This is the file that can be found on the name server disk and includes all resource records associated to the specific zone.

Normally are configured the content of the files, respective the content of the lines of the configuration files, including the Resolver file (the file of the Linux system on the side of the entries, in Linux server, from clients) normally the file : */etc/resolv.conf.*

The directories from the file */etc/resolv.conf* may to be configured with Linux system editors' commands, which open the respective file, and such as:
joe */etc/resolv.conf* (Enter)

Inside the file */etc/resolv.conf* each line (of command, which is not one Linux system command but is one line which indicate, as the content of one one directory, the behaviour of the DNS system) have one key word followed by one or many configuration values. The key commands: *nameserver* and *domain* accept a single value and *search, option* and *sortlist* accept one list of values. Look also to: FAQ of ActiveDomain.com / Domain Registration Service by Active-Domain Co. at: http://www.active-domain.com/help/faq-name-server.htm; http://en.wikipedia.org/wiki/Domain_Name_System etc.

The involved servers are:
- the Cache servers;
- the Slave servers and
- the Master server (the more complicated configuration works).

With the view of configuring of the Master server are used records such as SOA, NS, A, CNAME, MX and other, placed in the RRs- Resource Records, normally in the file *named.con* .
The file *named.conf* may to be open with the Linux system editor command, normally with the command:
 # joe /etc/other possible directories of the path /named.conf (Enter).

Start of Authority (SOA)

The SOA record (start of authority resource record), is very important, because it is the specific domain name official DNS record.

" an SOA record identifies the authoritative Name Server for a specific DNS database segment" http://en.wikipedia.org/wiki/SOA.
Practically SOA indicates the beginning of the data of one zone and defines the parameters of the respective zone.

The SOA record declares that the recorded resources which follow are the information with authority about the zone

It says the considered name server is the best information source of the zone. This name server will be the benchmark of "mysite.com" zone.
Inside a "zone file" there might be only one SOA record.
it may to read the resolver.config or other file, for instance with # **joe <respective name of the file including also the path toward the respective file>** (Enter), and where # is the Linux system prompt and the brackets <> must not be written inside the command

Considering examples previously mentioned we could have the following record:

```
@ IN SOA nameserver1.mysite.com. hostmaster.mysite.com. (
   20041014  ; serial number
   900 ; refresh
   600 ; retry
   86400 ; expire
   3600 ) ; default TTL
```

where
the symbol @ is referring at the current origin, in this case: nameserver1.mysite.com; 900 refresh time interval in seconds (at which the slave server refresh the Data from the Master server) ; 600 retry time interval (at which the slave to retry to accomplish the data from the master server, after one refuse of response) in seconds; 86400 expire time interval (the slave server waiting time in which the slave server will respond to the interrogations despite that has not the possibility of refreshment of the zone file; 3600 the Time to Live of the for the servers with negative cache.

Name Server Record (NS)

The Name Server Record (NS) is used to define the name servers to be employed for the domain.

Practically the NS identifies the name of the one domain server.

For each zone there must be at least a record of this type.

It might seem unnecessary to keep a resource record with DNS entries since it is kept by the registration agency record (where the domain look up phase starts).

This record is required if someone asks for the name servers of that domain.

NS records are mapped into canonical name (CN) type entries; in our case “nameserver1” and “nameserver2”:

mysite.com IN NS nameserver1.mysite.com
mysite.com IN NS nameserver2. mysite.com

In the above 2 lines, is declared that for the domain mysite.com are authoritative declared 2 name of servers for the respective domain

Hostname (A)

”A **hostname** (occasionally also, a **sitename**) is the unique name by which a network attached device (which could consist of a computer, file server, ….) is known on a network. The hostname is used to identify a particular host in various forms of electronic communication such as e-mail….” http://en.wikipedia.org/wiki/Hostname

The Address Record Entry (A) is added to the domain “forward lookup zone” and is associated to the IP address of the server that will manage requests for the specific domain.

Multiple records can be used; in that case requests assignment is made using “load balancing” technique (based on “round robin” algorithm) ”. **Round robin DNS** is a technique in which load balancing is performed by a DNS server instead of a strictly dedicated machine http://en.wikipedia.org/wiki/Round_robin_DNS .

In our example we could have to face the following situation:

```
mysite.com   IN  A   123.123.33.45.
mysite.com   IN  A   123.123.33.46.
```

In the above example, to the domain name mysite.com are assigned 2 IP addresses

Moreover it is possible to add this type of record for each machine inside the zone. For example:

```
myHost.mysite.com IN A 123.123.33.47
yourHost.mysite.com IN A 123.123.33.47
```

The using of the nslookup with the facility: set type

It is important to notice that previously described “nslookup” program allows collecting “Address” type records by default.

It is possible to use “set type” command, together with interactive modality, to change the type of resource record to interrogate.

There is a special type of query called “any” that allows collecting al resource records available for a specific host.

Furthermore you can use “server” command, which lets you set the used server as interrogations resolver. This capability is particularly useful to directly collect data from the name server delegated for the specific zone (known as “authoritative server”). In the following [Figure 13] an interactive mode utilization example is shown.

```
gino@fry:~$ nslookup
Default Server: mydns.mysite.com
Address: 1.1.1.1
>
> set type=NS
> yoursite.edu
Server: mydns.mysite.com
Address: 1.1.1.1
Non-authoritative answer:
yoursite.edu nameserver = dns1.yoursite.edu
yoursite.edu nameserver = dns2.yoursite.edu
yoursite.edu nameserver = dns3.yoursite.edu
Authoritative answers can be found from:
dns1.yoursite.edu inet address = 2.2.2.1
dns2.yoursite.edu inet address = 2.2.2.2
>
> server dns1.yoursite.edu
Default Server: dns1.yoursite.edu
Address: 2.2.2.1
>
> set domain=yoursite.edu
> set type=any
>
> yourhost
Server: dns1.yoursite.edu
Address: 2.2.2.1
yourhost.yoursite.edu inet address = 2.2.2.128
yourhost.yoursite.edu preference = 10, mail exchanger = yourhost.yoursite.edu
yourhost.yoursite.edu CPU=ALPHA OS=UNIX
>
> exit

gino@fry:~$
```

Figure 1.6 “nslookup” utilization example

In this example, at the beginning, the user asks for NS type records regarding “yoursite.edu” domain. From the data collected through this interrogation a server name is selected; requests will be sent to the same server. Afterwards the default domain is set using “set domain” command: “nslookup” command will use such information to expand hostnames during its interrogations.

In the above example is used not only the interactive action **exit** but also other interactive actions, such as **set**, may to be used

Next step consists on setting interrogation typology to “any” value, in order to avoid receiving only NS records.

In the end a particular host interrogation, inside default domain, is done.

The structure of the file /etc/hosts may to be viewed, in Linux, for example, with the command cat:

$ cat /etc/hosts (Enter)
#
#Table of IP Addresses and Hostames
#
127.0.1.1 localhost
172.5.5.5 bluebird
172.8.3.1 mary
……

In the above example the prompt is $

The character # indicates only the starting of the row with commentaries. >>

The command (in the following case, inLinux)
$ netstat –inet (Enter)

accomplishes and indicates on the screen the names of the hosts and the respective ports of the connection.>>

Canonical Names (CNAME)

The Alias Entry (CNAME) is a record that links an alias to a real name. For example:

www IN CNAME www.mysite.com

In the above example, the command said: the address www will be used as the address www.mysite.com .

Mail Exchanger (MX)

An **MX - Mail Exchange Record** is a type of resource record in the Domain Name System (DNS) specifying how Internet e-mail should be routed, respective identifies the e-mail Server for the respective domain.

Mail Exchanger (MX) configuration is required if a mail server for domain mail accounts management exists.

For example, an email address like “webmaster@mysite.com” requires a domain setting to resolve “mysite.com” mail server address. Such setup is very similar to CNAME one, but records will regard the mail server.

Multiple records of this type can exist:

mysite.com IN MX 10 mail.mysite.com

The above line of command (inside the RRs- Resource Records file), inside the configuration file, have precise made that the mail server mysite.com has the mail server mail.mysite.com and with the level of preference of the level 10. As the preference number is lower, the server is preferred

1.2.2. Web Server configuration

Once the DNS server has been installed and configured, wishing to send incoming requests to the corresponding IP address, it is necessary to configure the web server.

The web server needs to be properly configured to manage requests for the domain, either if they are based on the IP address or on headers including host names. The last ones are used by web servers in order to host multiple domains on a single IP address.

If using a Microsoft web server, Internet Information Server (IIS), it is necessary to create a new web site for the domain using IIS Manager. It is also required to add the domain (i.e. “mysite.com”) as a new host header that listens on the same IP address specified in the DNS entry. The port value is set to ‘80’, which is the default value for http requests. The host header can be added selecting “advanced” section near the web site IP address configuration.

Then the domain web site home directory must be set. Afterward it is necessary to add an additional host header for the “www”, i.e. “www.mysite.com”, so that everyone can have access to the web site typing “www” at the beginning of the URL.

If an Apache server is used, the domain will be configured through “VirtualHost” directives, located in “http.conf” configuration file.

The opening , viewing and editing of the respective file may to be achieved, for instance with the Linux system command joe: # **joe /etc/conf/httpd.conf** (Enter)

For example:

```
Listen 80
NameVirtualHost *

<VirtualHost *>
  ServerName www.mysite.com
```

```
  DocumentRoot /home/httpd/htdocs/
</VirtualHost>

<VirtualHost *>
  ServerName mysite.com
  DocumentRoot /home/httpd/htdocs/
</VirtualHost>
```

In the above text content of the directives are indicated the DNS names of the Virtua Host.
In the configuring text content of the directory is placed firstly the domain name www.mysite.com and in the second directive description mysite.com so that the server to may to be find with the both types of addresses

Hence domain configuration starts from domain registration authorized agency.
Such agency stores domain-name server correspondences.

The domain name resolution request, by designated DNSs, provides the correspondence between domain and web server IP address. At the end of this process, the web server satisfies requests basing on its domain specific configuration.

2. How to create web site examples for testing

The “httpd.conf file detailed description is discussed in other sections of this lesson [2]. previous lessons
Anyway, if we want to proceed with the following example, it is important to refer to it to notice that, if changing DocumentRoot (where HTML files and other files to be viewed on the Internet are located) location is required, it is possible to edit configuration file as follows:

```
DocumentRoot  /<pathVersoNostriDocs>/docs
```

Inside that directory we can insert all the resources that form our web site. For example, we can insert into it a set of HTML documents representing the data we want to collect via web.

Wishing to make the test extremely easy, we will just insert a HTML page containing classic example “Hello World!”. Such file will be called “hello.html” and will include the following piece of code:

```
<html>
   <head>
                <title> Hello World Example Page </title>
  </head>
  <body>
  <center>
                <h1> Hello, World! </h1>
                <p> This is a test web page </p>
                </center>
   </body>
</html>
```

After having rebooted Apache by instructions shown in the previous paragraph [], it will be possible to reach our site by the browser. Hence, typing the following URL in the browser tool bar:

```
http://myserver.mydomain/hello.html
```

we should be able to display our example page [Figure 14]

Figure 1.7 web page example

We need to follow another configuration step in order to make eventual CGI (Common Gateway Interface) scripts, included in our site, work.

3. Dynamic content

A CGI (Common Gateway Interface) script is a program written using one of many available languages (C/C++, Java, TCL, Perl, …).

CGIs are one of the modalities a web server can use to interact with external programs.

Usually such interaction has the aim of creating dynamic contents to be included into data provided by the web site.

The first step needed to test CGI scripts on our site is, first of all, configuring the Apache server to allow their execution.

There are many ways to create this configuration: we now assume to have previously created a directory including those scripts.

This directory must be pointed to the server through the configuration file using “ScriptAlias” directive.

In the following example we have put our scripts inside “cgi-bin” directory:

ScriptAlias /cgi-bin/ “<pathTowardsOurDocs>/cgi-bin”

Where “<pathTowardsOurDocs>” represents the location of the directory containing our CGIs.

In the above example, inside the file *httpd.conf* is inserted the directive which said: the /cgi-bin/ script (software program including lines based instructions) is placed at “<pathTowardsOurDocs>/cgi-bin”

CGI programs are often included, for security reasons, inside specific directories defined by the previous directive. In such way, webmasters can strictly check who is authorized to execute CGI programs.

However, if required security measures are adopted, there is no reason why CGI program cannot be run by an arbitrary directory.
This is the case in which, for example, we want to let any host user have its own CGIs: if we want to grant that property, without giving them an access to the main “cgi-bin” directory (Apache server one), it is necessary to let those programs be executed from other locations.

Wishing to do this, it is possible to use “Options” directive inside server configuration file. In our case:

<Directory “<pathVersoNostriDocs>/cgi-bin/">
Options ExecCGI

```
AddHandler cgi-script   .cgi   .pl
</Directory>
```

It is important to remember that the server is not executed by a privileged user.

Usually the process is run as “nobody” or “www” user and so extra permissions are required to execute some files on the server, like some CGI scripts, for example.

Usually, wishing to give required permissions to those files that must be executed from the “nobody” user, we must give every user the permission of executing files. For example:

```
chmod a+x myScript.pl
```

This is always true with the exception of the case in which the server is configured to use “suexec”.
The suexec is a software tool http://www.linuxplanet.com/linuxplanet/tutorials/1445/6/

The Syntax of the Linux command **chmod:**

 # chmod [-r] permissions filenames

“*-r* Change the permission on files that are in the subdirectories of the directory that you are currently in.
Permission Specifies the rights that are being granted. Below is the different rights that you can grant in alpha an numeric format.
Filename File or directory that you are associating the rights with.” http://www.computerhope.com/unix/uchmod.htm.

The command # **chmod a+x myScript.pl**
changes the permission of a file with the taking into consideration:

- the option **a** representing ALL;

the option **x** execute or run the file as a program.

This program allows CGI scripts execution with different permissions due to their location on the server.

“suexec” program has a very strict permission verification system and each failure of that system leads to a CGI program failure. (usually with an “Internal Server Error” message).
In this case it is necessary to consult the related log file to check which security check failed.

Simple program in CG

Moving back to our example we can try to write a simple CGI program.

There are two basic differences between a “normal” program and a CGI program.

The CGI program entire output must be preceded by a header defining the MIME-type, that is the HTTP header, which tells the client the type of content he is going to receive.

MIME- ***M****ultipurpose* ***I****nternet* ***M****ail* ***E****xtensions*, represents “a specification for formatting non-ASCII messages so that they can be sent over the Internet. Many e-mail-clients now support MIME, which enables them to send and receive graphics, , audio, and video files via the Internet mail system” http://www.webopedia.com/TERM/M/MIME.html

For example, if the output of our program must be an html page, the header will be:

```
Content-type: text/html
```

The second difference is based on the fact that the CGI program output must be in one of the formats supported by browsers (html, gif, etc.)

Except these two differences, writing a CGI program is not different from writing any other program.

In the following an example of CGI program written in Perl is shown. We assume that the software required to execute Perl scripts has been previously installed on the server.

The program just writes a HTML page to be displayed in the browser window.

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html> <head>";
print "<title>Hello World Example Page 2 </title>";
print "</head><body>";
print "<center>";
print "<h1> again ... Hello, World! </h1>";
print "<p> This is a dynamic web page </p> ";
print "</center>";
print "</body> </html>";
```

The first line tells the server, or in general to the shell that executes it, that the program can be executed giving the file to the /usr/bin/perl interpreter.

The second line prints the content-type, followed by two 'return" characters: in HTTP protocol semantics this defines the header end and the beginning of the reply body.

Remaining lines represent the code we want to display on the client browser, which, theoretically, could be obtained by a much more complex elaboration.

Once previous commands have been saved to a file, i.e. "hello.pl", it is necessary to move it to the "cgi-bin" directory.
The moving to the cgi-bin directory may to be achieved, for instance, with the Linux command move:
 # mv hello.pl /<to the path, indicated in the cgi-bin directory, as it is described in the httpd.conf file>/ (Enter); where the text between the brackets willl be replaced with real path

Opening the browser and typing the following URL

http://myserver.mydomain/cgi-bin/hello.pl

it should be possible to display, inside the browser window, the page dynamically generated [Figure 2.1]

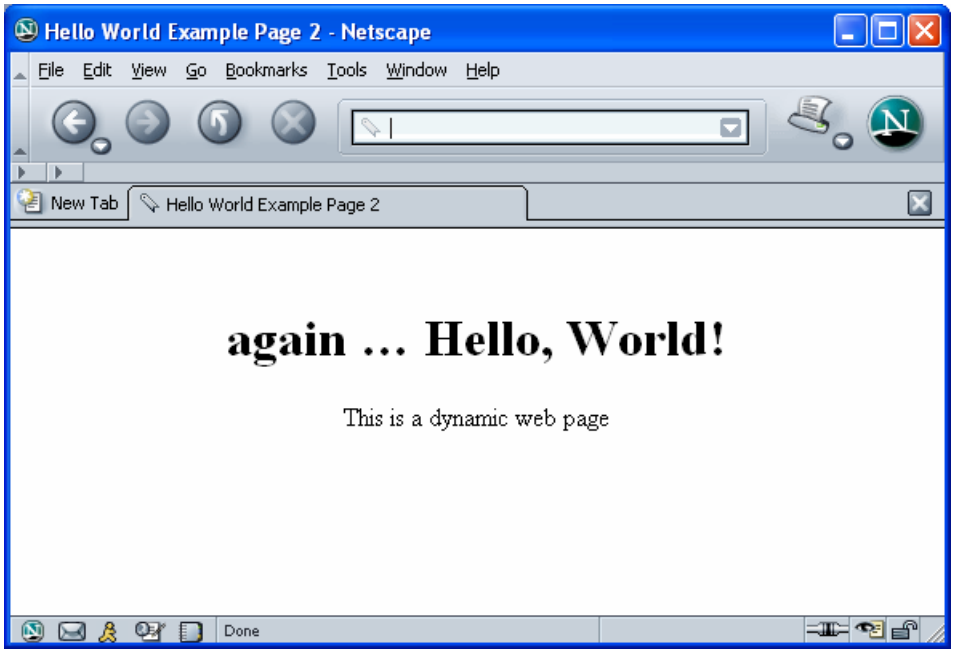

Figure 2.1 example output CGI script

4. How to let the owner of the hosted website administrate web facilities

In the first years of the web, sites were sets of pages basically static, interconnected by links.

They were typically realized and kept by technical personnel that knew HTML language: the web contents production language.

Usually sites were realized basing on a starting project and updated by the same people they had been realized by, so the evolution was in the hand of the developer.

Modifies, requested for different reasons, were formalized by marketing or commercial offices, and passed to developers, which modified pages.

Hence site modification, based on client needs, was made page by page by technical attendants.

The following step has been the achievement of dynamic sites.

In practice it is a fixed structure, in which dynamically created contents are inserted basing on specific requests. Such contents can be read from a database or be the result of a more complex elaboration.

The advantage consists on the fact that the site structure remains logically separated from the content, which can be easily updated basing on owner needs.

For example, a card with a fixed structure and representing a phonebook can be filled from time to time with specific required record data.

Comparing this technique to the first one (static one) it easy to see many advantages:

- Simple maintenance;
- Less pages required;
- Possibility of modifying contents without modifying pages structure.

The management of contents that must be shown through a dynamic site is usually made by a Content Management System (CMS).

Moreover, it is important to notice that in actual e-business based world contents assumed a fundamental relevance.

Many organizations have web site or are about to make one. If a company is not able to update continuously data regarding its products, then it will not be able to satisfy its customers' expectations.

Content Management Systems (CMSs) can be a great help for contents supply computerization, in order to enhance contents flow without investing great amounts of money.

The term "content", in a web site context, includes software entities like images, documents (reports, fact sheets etc.), audio and videos.

The adoption of a contents management system can be justified in one or more of the following cases:

- When web publication is divided into different company locations and communicating contents variations can be extremely difficult;
- The web site is wide and frequent updates are required;
- The company has customers that actively contribute to site contents provision;
- The inner circle of personnel has the entire control of the site management under an organizational and technical point of view.

These types of application allow, after having created a structure dynamically producing pages, site contents modification without required technical development knowledge.
In this way the marketing responsible or the commercial office responsible can insert data reflecting contents to be modified, without the necessity of any communication to the technical personnel.

The site owner can therefore modify the site itself without modifying a big amount of pages.

The CMS main function consists on managing contents during all their lifetime, that is from their creation to their publication and successive removal.

Practically a Content Management System is a combination of databases, file systems and other software used to store and afterwards reuse great amounts of data.

Trying to increase detail level, a Content Management System is a software application, typically usable through a browser, that allows inserting and managing contents to be shown to the site users. That management is done by a user interface extremely user-friendly.

The contents managing system can therefore be seen as a set of software modules that work together to provide a flexible site management. In the following, main modules are shown:

- The database with the contents;
- The pages publishing engine;
- The real contents management system.

The database is the archive of the contents that will be published on the site.
Usually it includes the dynamic web site part, the content managed by the user, visibility rules and access data.

The pages publication engine is the part (of the entire software) dedicated to contents publication.
Basically it is a software application able to update the site pages.

Finally the contents manager is the front end used by the operator to manage site contents.
Functionalities of that module can be generally summarized as editor functionalities, for pages to be produced, and as file manager ones (for visibility management).

It is worth noticing that usually this part of the software presents some differences: actually it is possible to move from applications allowing simple text introduction to the ones that provide graphic interfaces and visibility rules depending on the user.

Contents related data, managed by the management system, are usually inserted into a database, and afterwards taken by the dynamic engine in order to produce pages to be sent to the client.

Inside the contents management system there are typically three types of users:

- Contents editors;
- Publishers;
- Authors.

The first ones decide which contents will be published and where, the second ones follow the publication process and the authors create the site content.

Therefore in this sense a CMS allows authors and editors, lacking in advanced technical abilities, publishing fast and easily their content, that otherwise would have been managed by programmers.

In this way developers can concentrate on site planning, focusing on the site structure, on browsing aspects and, in general, on technical aspects, without paying attention to contents creation and maintenance.

The status of the entire site can remain consistent, because the content is taken from the database and automatically inserted into previously created (by developers) templates.

On the contrary, authors can focus on contents writing without worrying about planning matters.

Contents can be easily added to the database and displayed on web pages, inserting pieces of text inside the fields of a form in the administration system of the CMS.

Data related to the content inserted, as titles, descriptions, keywords, author, publication date etc, can be inserted with the same contents. This allows a better contents management and updating.

The system manages publication processes and the rights of its users towards such processes.

That is it is able to manage workflow and quality control, letting a system administrator configure roles and permissions for contents authors, editors and publishers.

For example, many authors can be authorized to insert new material on the contents management system.

The editor can be alerted, by a notification system, as a consequence of such addition. The new material can therefore be revisioned, approved and finally published on the web by authorized personnel by a button click.

Hence, using these functionalities, the company can save on training periods and allow more people to publication.

The CMS reduces therefore publication required periods. This is an extremely important feature for modern companies; the faster contents are published, the bigger is the created value. A wide range of contents can be published by a CMS, for example;

- Simple pages;
- Complex pages, with specific layouts;
- Dynamic information generated by a database;
- Training material;
- Online manuals;
- documents

Using technologies like XML (eXtensible Markup Language), a contents management system is able to provide the required abstraction level of the data that must be presented to users to grant data portability.

Such portability is concretised, for example, in the service itself provision to mobile telephones, palm tops and TV, as well as to PCs.

The necessity of a company and of its web site, define features required to a contents management system.

Anyway, some important features to be considered are listed below:

- Template web pages: web pages models must be changeable and pursuant to actual standards related to web technologies. All of this is done to assure maximum compatibility between browsers and different platforms.
- Access permissions management: it would be desirable that only authorized users could publish material on the site, once the content has been approved.
- Possibility of integration with legacy systems.
- Contents creation and publication without the need of writing pieces of code.
- Management of inserted content (author, update date, etc.) additional data (metadata)

There are open source CMSs or commercial solutions. The choice tightly depends on the requirements of the company that wants to own a contents management system.

Key Point Summary Conclusions and Recommendations

1. In the fisrt section the name server configuration process is briefly introduced, with the aim of making the web site available by a domain name instead of its IP address. After an overview on Domain Name System functioning, service creation main steps are shown. These steps are:
- Domain name registration;
- Choice and configuration of the server that will be used as name server;
- Web server configuration.
2. The second section is related to a simple web site (made by an HTML page and a CGI script) creation and deployment process. These examples are afterwards inserted in the web server configuration so to be used by a normal web browser.
3. The last section of the lesson introduces concepts related to web site content management. By the evolution of the functionalities provided by web servers and their increasing complexity, content management tools allow a content administration independent from the site structure (highly linked to programming). In this way a user lacking in programming skills can modify and, in general, manage the web site content.

Study Guide

ESSENTIAL QUESTIONS TO VERIFY THE ACCOMPLISHED KNOWLEDGE

1. How a DNS can be defined
2. How a domain can be registered.
3. Record types included in the DNS

BIBLIOGRAPHY. REFERENCES.

[1.] R.Allen, M.Larson, C.Liu, "*DNS on Windows Server 2003*", O'Reilly, Dec 2003, ISBN 0-596-00562-8
[2.] C.Hunt, "*TCP/IP Network Administration*", O'Reilly, Dec 1997, ISBN 1-56592-322-7
[3.] R.Stevens, "TCP-IP Illustrated Volume 1"
[4.] D.Heywood, R,Scrimger, "*Networking with Microsoft TCP/IP…*", New Riders Pub., 1997, ISBN 1-56205-791-X
[5.] Craig Hunt: *Linux DNS Server Administration*, Sybex Inc., 2000; 0782127363

SUPPLEMENTARY IMPORTANT BLIOGRAPHY. REFERENCES. www

[Sup.1] http://www.linux-mag.com/content/view/233/2083/ Linux Magazine, Open Sources, Open Standards. © 1999-2005 Infostrada Communications, LLC.

SUPPLEMENTARY INDICATIONS ABOUT THE CONTENT OF THE LESSON

It is recommended to consult also: *Domain Name System* at Wikipedia: http://en.wikipedia.org/wiki/CNAME

RESPONSES TO THE QUESTIONS

1. The Domain Name System (DNS) is a distributed database. Particular applications, called Name Servers, are the server part of the client-server architecture on which DNS is based

2. Each domain name is registered through an agency by ICANN (Internet Corporation for Assigned Names and Numbers). It is a no-profit international agency whose main tasks consist on assigning IP (Internet Protocol) addresses and managing Top-Level and generic level (gTLD) domain name system and international codes (ccTLD) system.

3. The main concept at the base of the domain name is the creation of a "zone" where to search for resolution. Inside that "zone" the following entries are added:

- Start of Authority (SOA);
- Name Server (NS);
- Hostname (A);
- Canonical Names (CNAME);
- Mail eXchange (MX).

WORDS TO THE LEARNER: <<"*Do not wait for opportunities. Create them*" (After Bernard Show)>>.