

How To Boot And Then Verify That A Linux Host Is Available On The Internet

Graham Leach, Partner

TIG – The Imperators Group

Copyright (c), 2002

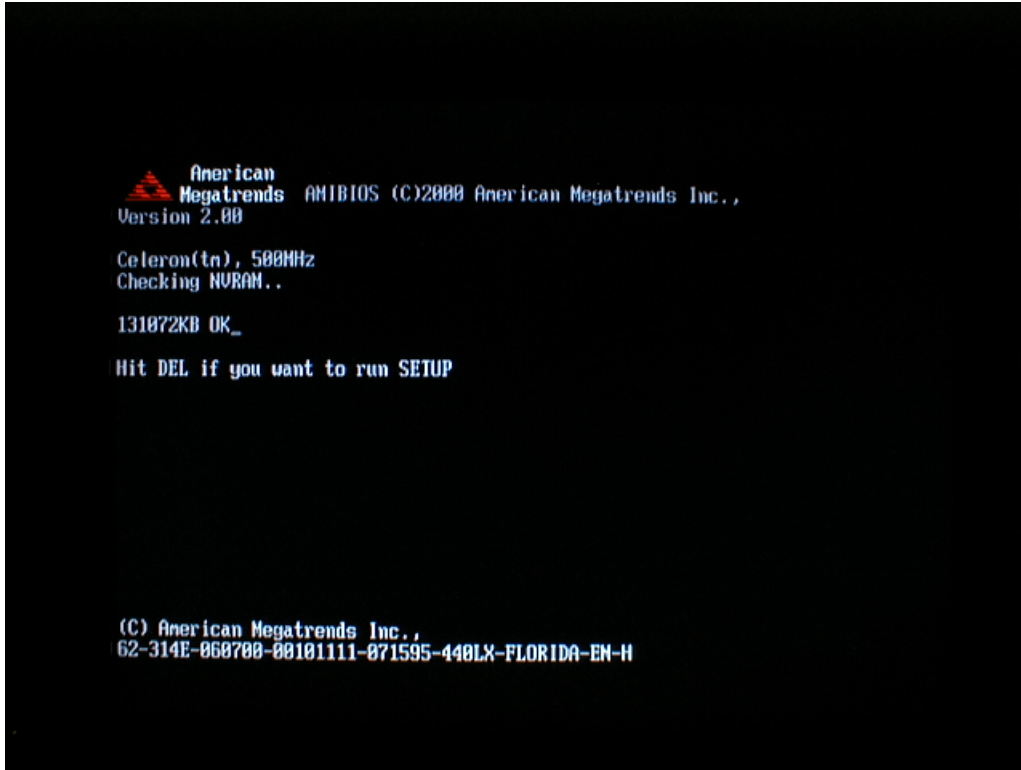
Table Of Contents

Booting a Linux Host.....	3
The BIOS Setup Screen	4
Getting Into the IDE Auto Detect Screen	5
Checking the results of the IDE Auto Detect Screen	6
Saving BIOS Changes.....	7
The Boot Screen	8
Watching The Kernel Boot.....	9
The Red Hat Login Screen	10
Checking That The Host Is Up On The Internet.....	11
Checking /etc/resolv.conf	12
Checking /etc/hosts	13
Testing Loopback Networking	14
Testing Primary Interface Networking	15
Testing Name Resolution With nslookup	16
If Every Step Was Successful Then This System Is UP!.....	16

Booting a Linux Host

Typically when a Linux host becomes available to boot it is when the machine has been turned on

Consider the following.

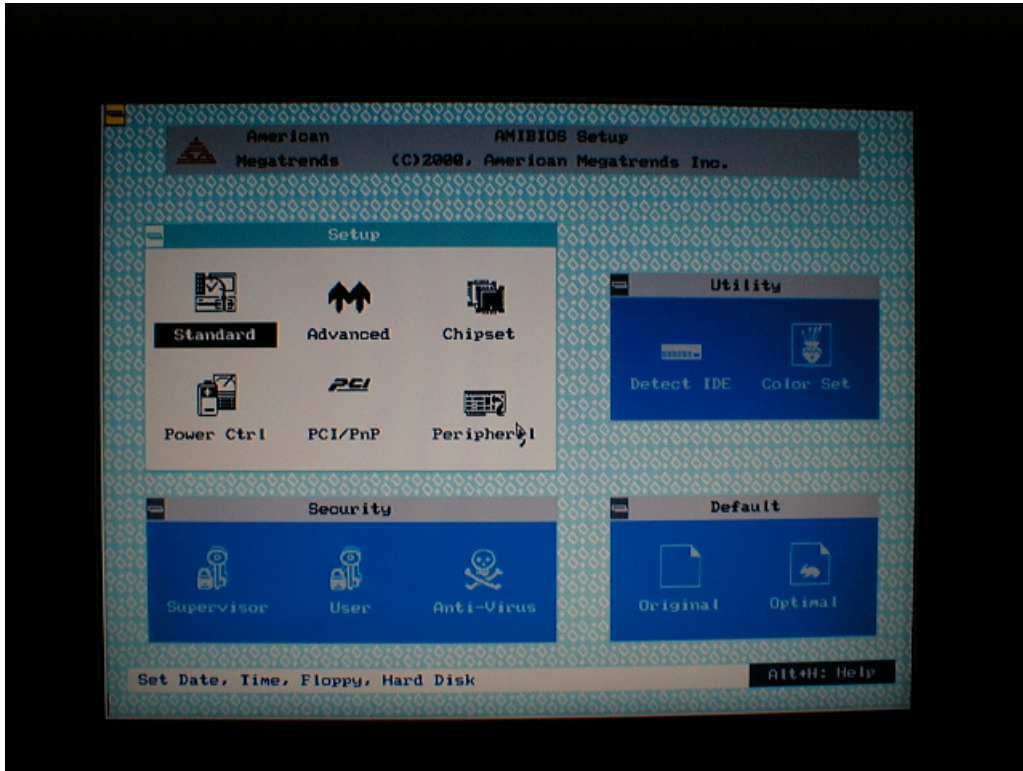


In the example, the machine is at the very beginning of its boot cycle. The memory of the machine has been counted and the BIOS prompt is displayed.

If there is a need to adjust the BIOS setting of the computer, at this time it is necessary to press the **DEL** key. If there is no need to adjust the BIOS it is not necessary to press the **DEL** key.

The BIOS Setup Screen

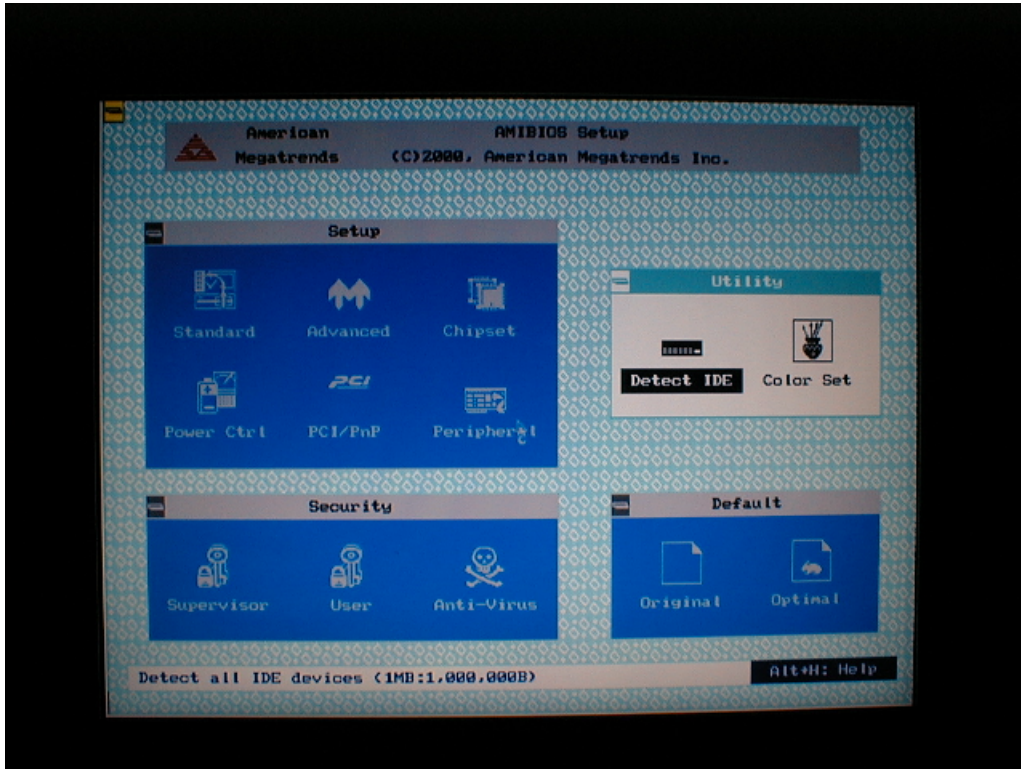
Consider the following:



In the above example, the BIOS has been entered and the initial screen is being displayed. If there is a need to adjust settings in any screen other than the main one (upper left corner) the <TAB> key must be pressed.

Getting Into the IDE Auto Detect Screen

Consider the following:

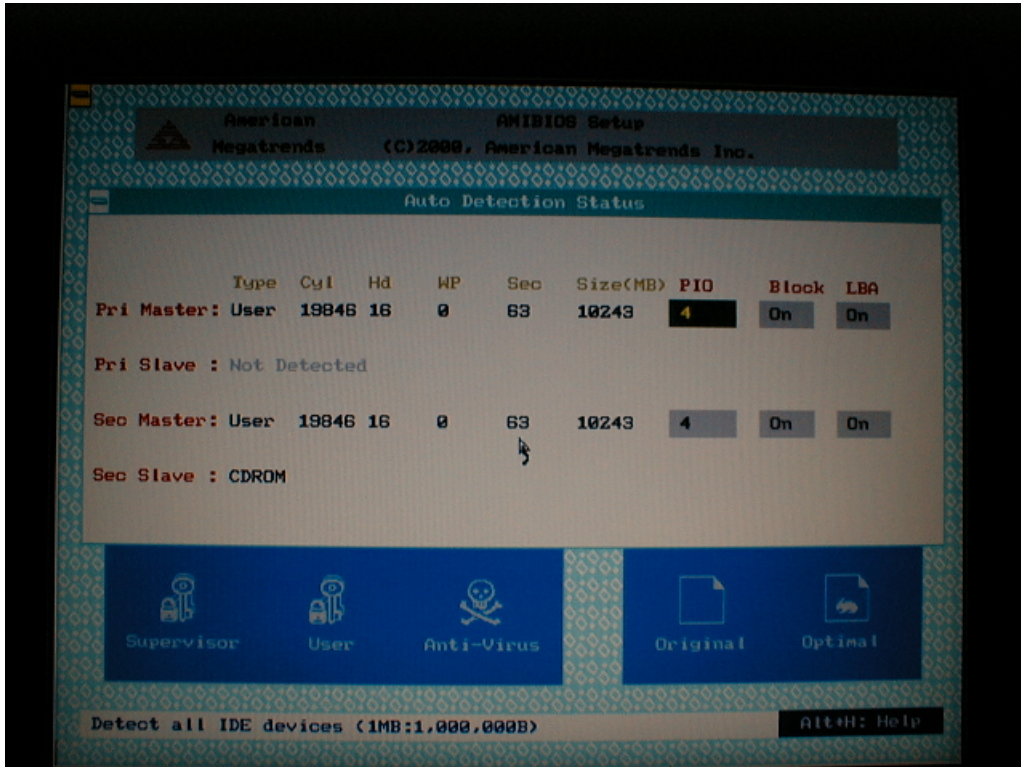


In the above example we are on the upper-right menu and about to invoke the **Detect IDE** utility. This utility will probe the hard drives and obtain their geometry information.

Run the utility by pressing the <ENTER> key.

Checking the results of the IDE Auto Detect Screen

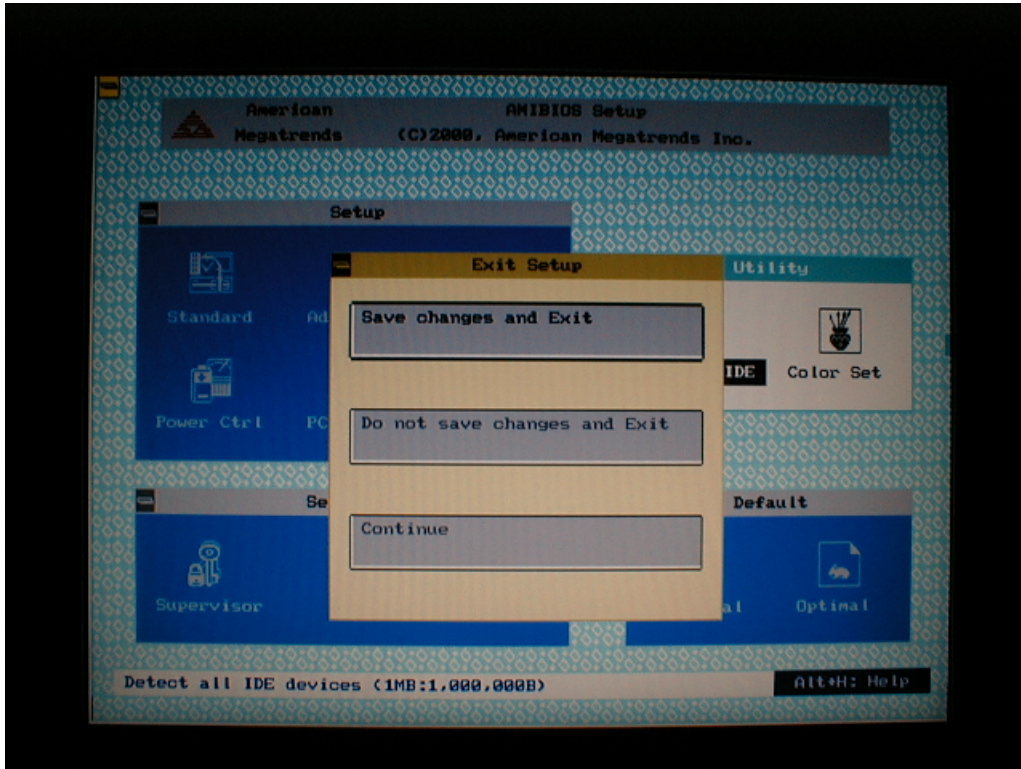
Consider the following:



In the above example the **Detect IDE** utility has been run.

Saving BIOS Changes

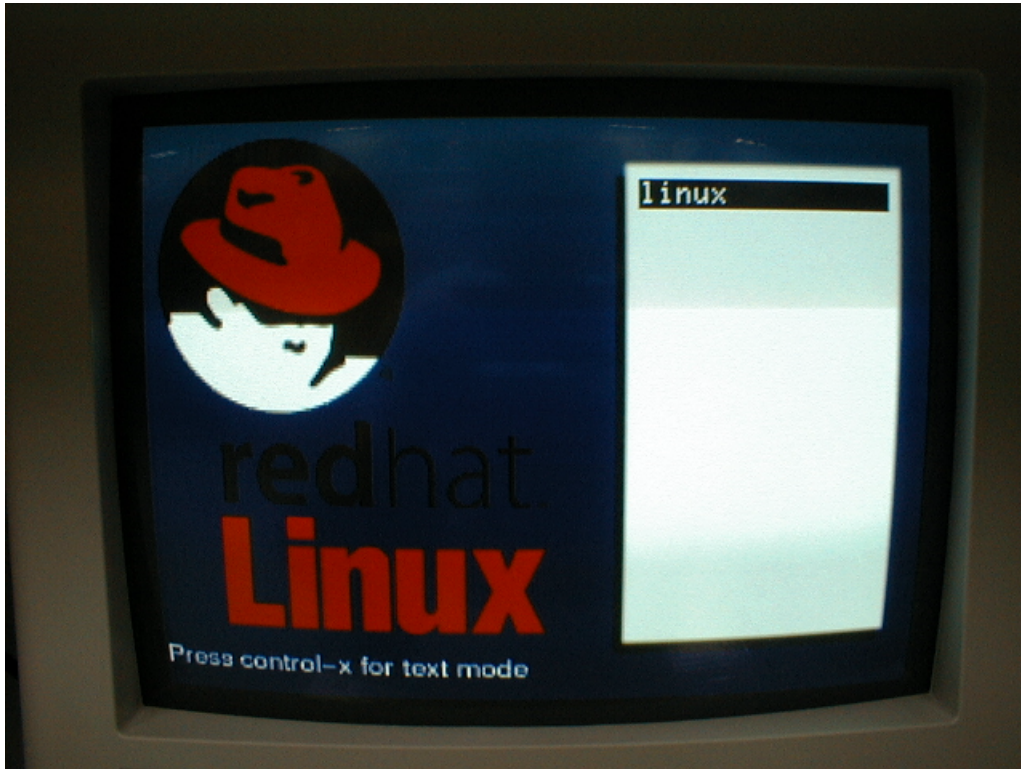
Consider the following:



After the BIOS settings have been verified (and written down) they are saved to NVRAM and the machine rebooted.

The Boot Screen

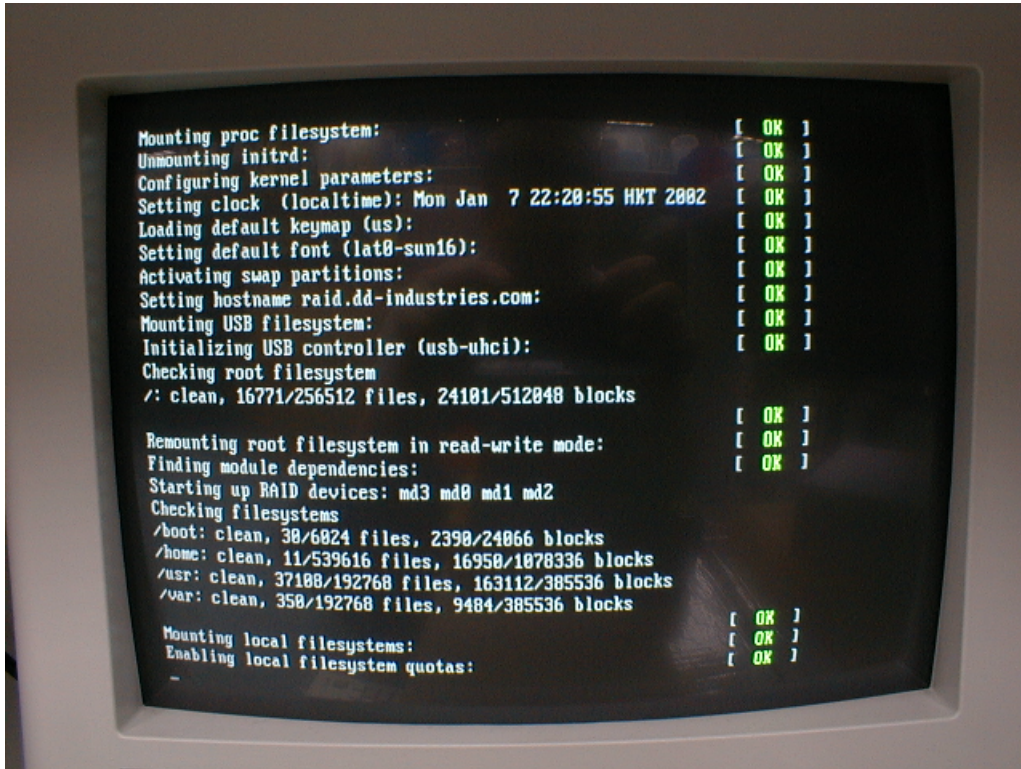
Consider the following:



In the example the machine is ready to boot. Press <ENTER> to boot the kernel, or just wait an appropriate amount of time and the system will boot automatically.

Watching The Kernel Boot

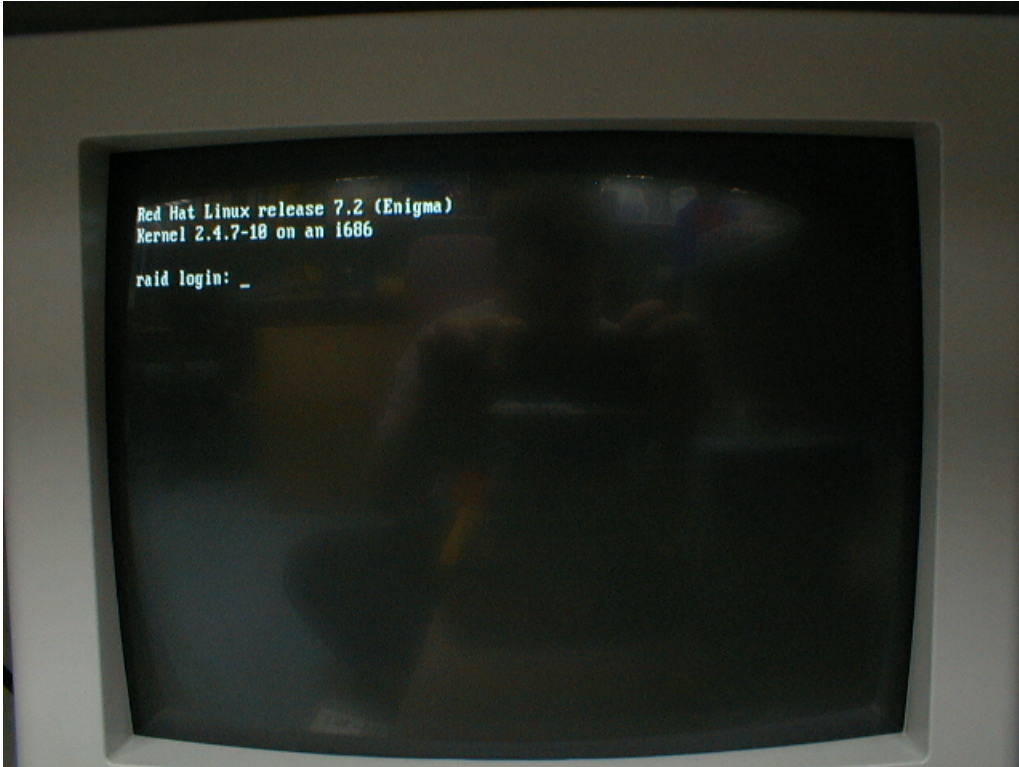
Consider the following:



As the kernel boots it shows systems coming up. If a given subsystem activates without any problems a **[OK]** message is displayed. If there is a problem, a **[FAILED]** message appears.

The Red Hat Login Screen

Consider the following:



In the example the linux host has booted and it is now necessary to input the user credentials for **root**.

Checking That The Host Is Up On The Internet

Consider the following:

```
root@ddi01.dd-industries.com: /root
[root@ddi01 /root]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:B4:5E:36:9E
          inet addr:203.80.245.1 Bcast:203.80.245.7 Mask:255.255.255.248
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2828380 errors:1 dropped:124 overruns:0 frame:47
          TX packets:2992094 errors:0 dropped:0 overruns:0 carrier:0
          collisions:18279 txqueuelen:100
          Interrupt:11 Base address:0xe000

eth1      Link encap:Ethernet  HWaddr 00:50:FC:2A:C3:6C
          inet addr:192.168.0.1 Bcast:192.168.0.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3661169 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3150391 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0xe400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:8443 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8443 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

[root@ddi01 /root]#
```

In the example the **ifconfig** command has been issued to verify that the Ethernet interface(s) are working properly and have been assigned an IP address.

As can be seen on the screen, there are **three** interfaces on this computer:

<i>Device</i>	<i>Description</i>	<i>IP Address</i>	<i>STATUS</i>
eth0	Primary Interface	203.80.245.1	UP BROADCAST RUNNING
eth1	Secondary Interface	192.168.0.1	UP BROADCAST RUNNING
lo	Loopback Interface	127.0.0.1	UP LOOPBACK RUNNING

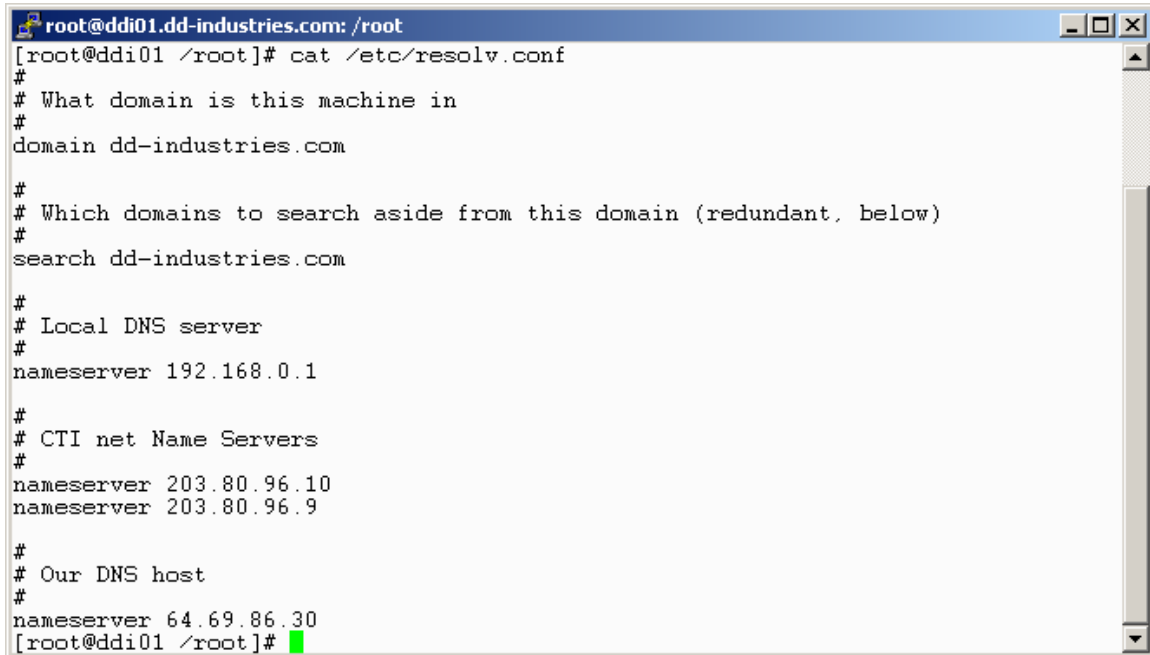
On other computers there may be more or less interfaces, but there must be a **minimum** of two basic interfaces for the Linux host to work properly:

<i>Device</i>	<i>Description</i>	<i>IP Address</i>	<i>STATUS</i>
eth0	Primary Interface	203.80.245.1	UP BROADCAST RUNNING
lo	Loopback Interface	127.0.0.1	UP LOOPBACK RUNNING

The **eth0** interface is the interface that is normally used to communicate with the Internet, the **lo** interface is often used by the Linux kernel or programs to allow communications within the computer itself.

Checking /etc/resolv.conf

Consider the following:

A terminal window titled 'root@ddi01.dd-industries.com: /root' showing the output of the command 'cat /etc/resolv.conf'. The output displays the configuration for the /etc/resolv.conf file, including domain, search, and nameserver settings.

```
root@ddi01.dd-industries.com: /root
[root@ddi01 /root]# cat /etc/resolv.conf
#
# What domain is this machine in
#
domain dd-industries.com

#
# Which domains to search aside from this domain (redundant, below)
#
search dd-industries.com

#
# Local DNS server
#
nameserver 192.168.0.1

#
# CTI net Name Servers
#
nameserver 203.80.96.10
nameserver 203.80.96.9

#
# Our DNS host
#
nameserver 64.69.86.30
[root@ddi01 /root]#
```

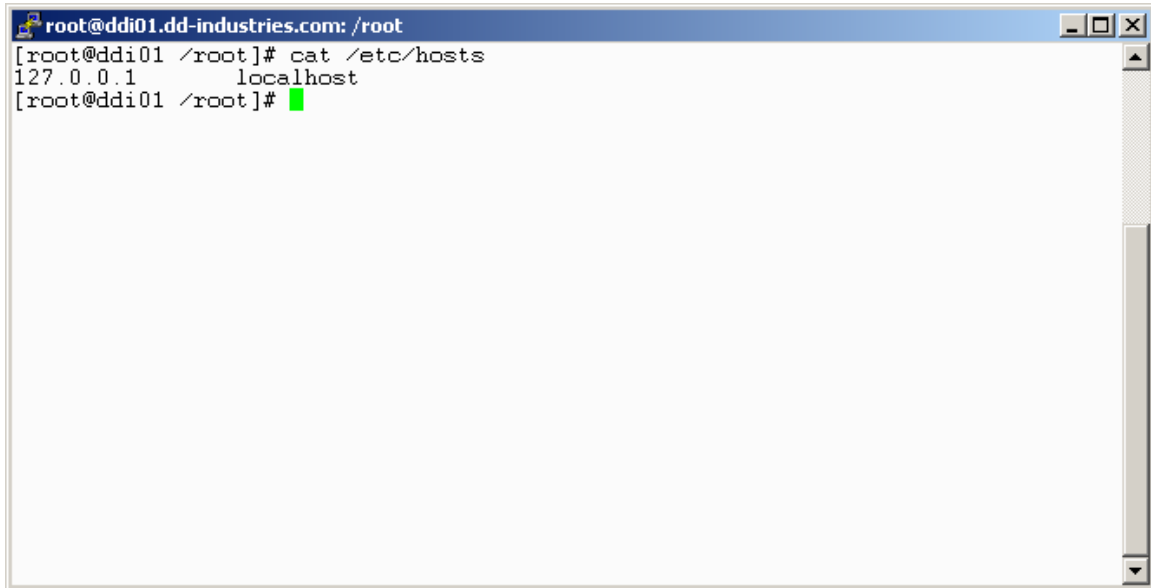
In the example the **/etc/resolv.conf** file has been output to the screen.

The file **/etc/resolv.conf** is used to control how a linux computer resolves name queries. As can be seen within the output, many settings are controlled within this file:

Entry	Description
domain	What domain name to “tack on” to the end of incomplete host names (e.g. ddi02)
search	Additional domains to search if the primary domain does not render a result
nameserver	The name servers for this host

Checking /etc/hosts

Consider the following

A terminal window with a blue title bar containing the text 'root@ddi01.dd-industries.com: /root'. The terminal shows the command 'cat /etc/hosts' being executed, with the output '127.0.0.1 localhost' displayed on the next line. The prompt '[root@ddi01 /root]#' is shown again on the third line, followed by a green cursor.

```
root@ddi01.dd-industries.com: /root
[root@ddi01 /root]# cat /etc/hosts
127.0.0.1      localhost
[root@ddi01 /root]#
```

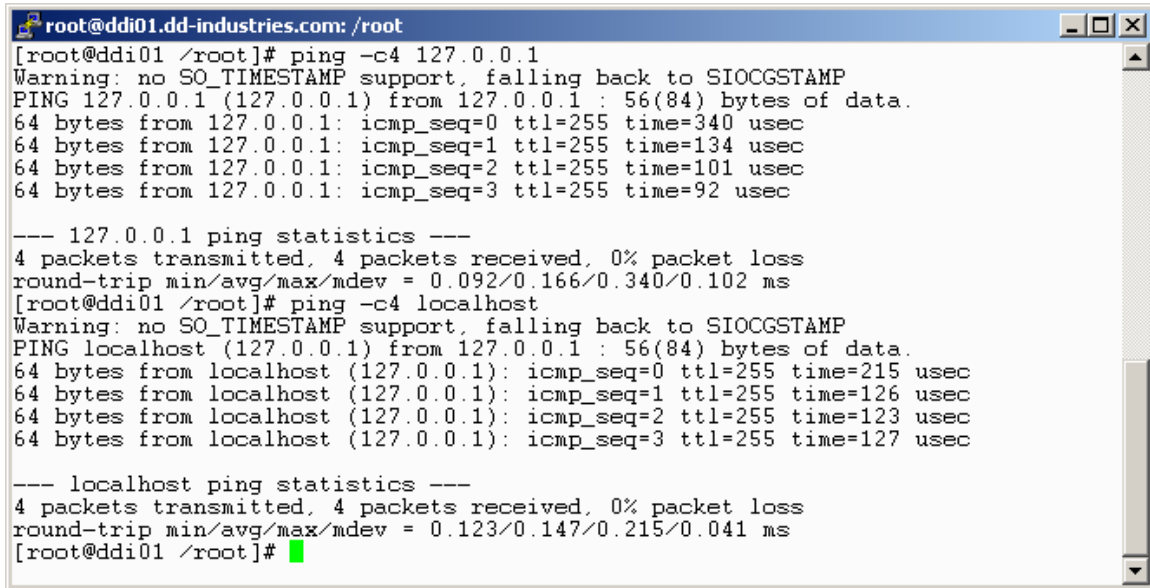
In the example the **/etc/hosts** file has been output to the screen. Before the system has DNS working (or even networking up) it sometimes needs to define some things on an IP level in terms of host -> IP mapping.

The **/etc/hosts** file pre-dates DNS and was the original way that hostname to IP address conversion was accomplished. This was replaced by DNS because it wasn't scalable.

The thing to check here is that **localhost** has been mapped to the IP address **127.0.0.1**

Testing Loopback Networking

Consider the following

A terminal window titled 'root@ddi01.dd-industries.com: /root' showing the results of two ping commands. The first command is 'ping -c4 127.0.0.1', which shows four successful pings with decreasing times (340, 134, 101, 92 usec) and a summary showing 0% packet loss. The second command is 'ping -c4 localhost', which also shows four successful pings with times (215, 126, 123, 127 usec) and a summary showing 0% packet loss.

```
root@ddi01 /root]# ping -c4 127.0.0.1
Warning: no SO_TIMESTAMP support, falling back to SIOCGSTAMP
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=340 usec
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=134 usec
64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=101 usec
64 bytes from 127.0.0.1: icmp_seq=3 ttl=255 time=92 usec

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.092/0.166/0.340/0.102 ms
root@ddi01 /root]# ping -c4 localhost
Warning: no SO_TIMESTAMP support, falling back to SIOCGSTAMP
PING localhost (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=0 ttl=255 time=215 usec
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=255 time=126 usec
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=255 time=123 usec
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=255 time=127 usec

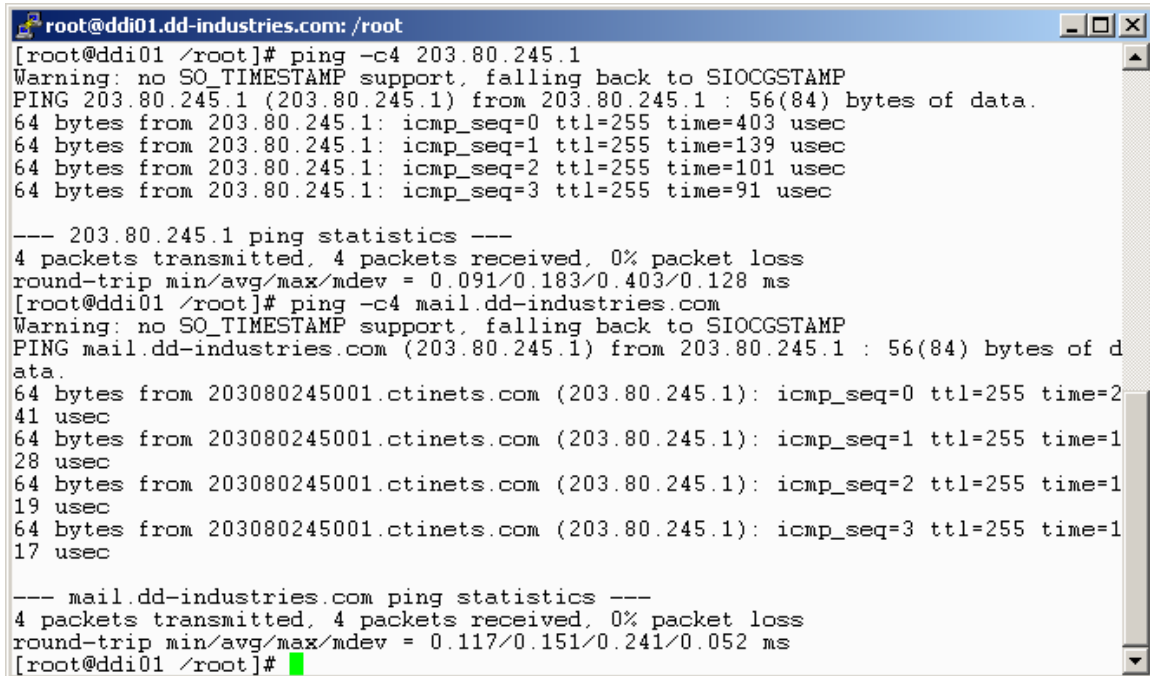
--- localhost ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.123/0.147/0.215/0.041 ms
root@ddi01 /root]#
```

In the example the **loopback** address was tested in two ways. First of all the raw IP address of **127.0.0.1** was used to make sure that the Linux kernel was handling TCP/IP properly.

Next the name **localhost** was used to test the name -> IP mapping of the **/etc/hosts** file.

Testing Primary Interface Networking

Consider the following:



```
root@ddi01.dd-industries.com: /root
[root@ddi01 /root]# ping -c4 203.80.245.1
Warning: no SO_TIMESTAMP support, falling back to SIOCGSTAMP
PING 203.80.245.1 (203.80.245.1) from 203.80.245.1 : 56(84) bytes of data.
64 bytes from 203.80.245.1: icmp_seq=0 ttl=255 time=403 usec
64 bytes from 203.80.245.1: icmp_seq=1 ttl=255 time=139 usec
64 bytes from 203.80.245.1: icmp_seq=2 ttl=255 time=101 usec
64 bytes from 203.80.245.1: icmp_seq=3 ttl=255 time=91 usec

--- 203.80.245.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.091/0.183/0.403/0.128 ms
[root@ddi01 /root]# ping -c4 mail.dd-industries.com
Warning: no SO_TIMESTAMP support, falling back to SIOCGSTAMP
PING mail.dd-industries.com (203.80.245.1) from 203.80.245.1 : 56(84) bytes of data.
64 bytes from 203080245001.ctinets.com (203.80.245.1): icmp_seq=0 ttl=255 time=241 usec
64 bytes from 203080245001.ctinets.com (203.80.245.1): icmp_seq=1 ttl=255 time=128 usec
64 bytes from 203080245001.ctinets.com (203.80.245.1): icmp_seq=2 ttl=255 time=119 usec
64 bytes from 203080245001.ctinets.com (203.80.245.1): icmp_seq=3 ttl=255 time=117 usec

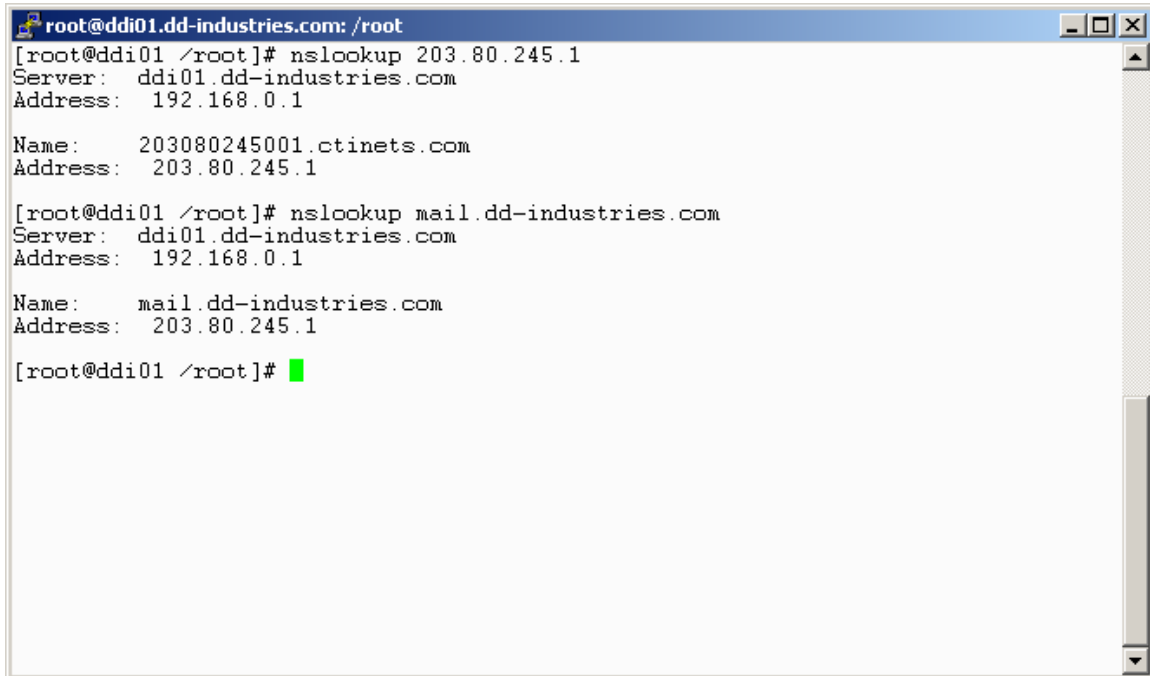
--- mail.dd-industries.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.117/0.151/0.241/0.052 ms
[root@ddi01 /root]#
```

In the above example the raw IP address was used to test if the interface was processing packets correctly by using the **ping** command to send TCP/IP packets to the Ethernet Interface at **eth0**, which has been assigned the IP address **203.80.245.1**.

Next the **Fully Qualified Domain Name (FQDN)** of the computer was used to test if the DNS resolution was working correctly for this computer.

Testing Name Resolution With nslookup

Consider the following:



```
root@ddi01.dd-industries.com: /root
[root@ddi01 /root]# nslookup 203.80.245.1
Server: ddi01.dd-industries.com
Address: 192.168.0.1

Name: 203080245001.ctinets.com
Address: 203.80.245.1

[root@ddi01 /root]# nslookup mail.dd-industries.com
Server: ddi01.dd-industries.com
Address: 192.168.0.1

Name: mail.dd-industries.com
Address: 203.80.245.1

[root@ddi01 /root]#
```

In the example the **nslookup** command was used to perform a **reverse lookup** and a **forward lookup**.

Reverse lookups attempt to obtain the Fully Qualified Domain Name (FQDN) of an IP address, **forward lookups** attempt to obtain the IP address of a supplied FQDN.

If Every Step Was Successful Then This System Is UP!