Assignment No: 1 Explore and Study of TCP/IP utilities and Network Commands on Linux.

a) Ping g) Tracert/Traceroute/Tracepath b) ipconfig / ifconfig h) NSlookup c) Hostname i) Arp

d) Whois j) Finger e) Netstat k) Port Scan / nmap f) Route

Theory:

a) **Ping:**
The ping command is used to check if the remote server is reachable or not. Ping command has the following syntax.
Ping Using DNS Name

```
1 ubuntu@devopscube:~$ ping devopscube.com
2 PING devopscube.com (208.91.198.132) 56(84) bytes of data.
3 64 bytes from cp-18.webhostbox.net (208.91.198.132): icmp_seq=1 ttl=51 time=64.8 ms
4 64 bytes from cp-18.webhostbox.net (208.91.198.132): icmp_seq=2 ttl=51 time=65.3 ms
```

Ping Using The IP Address

```
1 ubuntu@devopscube:~$ ping 8.8.8.8
2 PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
3 64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=8.47 ms
4 64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=8.68 ms
```

b) **Limit The Packets To Send**

c) If you want to limit the ping output without using ctrl+c, then you can use the "-c" flag with a number as shown below.

```
1 ubuntu@devopscube:~$ ping -c 1 devopscube.com
2 PING devopscube.com (208.91.198.132) 56(84) bytes of data.
3 64 bytes from cp-18.webhostbox.net (208.91.198.132): icmp_seq=1 ttl=51 time=64.8 ms
4 --- devopscube.com ping statistics ---
5 1 packets transmitted, 1 received, % packet loss, time 0ms
6 rtt min/avg/max/mdev = 64.885/64.885/64.885/0.000 ms
7 ubuntu@devopscube:~$
```

b) Ip (Ifconfig)

ip command is used to display and manipulate routes and network interfaces. **ip**command is the newer version of **ifconfig.** ifconfig works in all the systems, but it is better to use ip command instead of ifconfig. Let's have a look at few examples of ip command.

Display Network Devices And Configuration

```
 1  ip a or ip addr
1  ubuntu@devopscube:~$ ip a
2
3  1: lo: &lt;loopback,up,lower_up&gt; mtu 65536 qdisc noqueue state UNKNOWN group default
4   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
5   inet 127.0.0.1/8 scope host lo
```

```
6   valid_lft forever preferred_lft forever
7   inet6 ::1/128 scope host
8   valid_lft forever preferred_lft forever
9  2: eth0: &lt;broadcast,multicast,up,lower_up&gt; mtu 9001 qdisc pfifo_fast state UP group default qlen
10 1000
11  link/ether 06:e1:76:a0:eb:c9 brd ff:ff:ff:ff:ff:ff
12  inet 172.31.18.184/20 brd 172.31.31.255 scope global eth0
13  valid_lft forever preferred_lft forever
    inet6 fe80::4e1:76ff:fea0:ebc9/64 scope link
    valid_lft forever preferred_lft forever
```

You can use this command with pipes and grep to get more granular output like IP address of eth0 interface. The following command gets the IP address of eth0 network interface.

```
ip a | grep eth0  | grep "inet" | awk -F" " '{print $2}'
```

```
1 ubuntu@devopscube:~$ ip a | grep eth0  | grep "inet" | awk -F" " '{print $2}'
2 172.31.18.184/20
3 ubuntu@devopscube:~$
```

c) Traceroute

The traceroute command shows how a data transmission travelled from a local machine to a remote one. A typical example would be loading a web page. Loading a web page over the internet involves data flowing through a network and a number of routers. The traceroute command can show the route taken and the IP and hostnames of routers on the network. It can be useful for understanding latency or diagnosing network issues.

```
1 syntax: traceroute
2 traceroute google.com
  traceroute to google.com (173.194.33.163), 30 hops max, 60 byte packets
1  1  ec2-50-112--84.us-west-2.compute.amazonaws.com (50.112.0.84)  1.974 ms  1.895 ms  1.899 ms
2  2  100.64.1.247 (100.64.1.247)  1.414 ms 100.64.1.137 (100.64.1.137)  1.127 ms 100.64.1.97
3 (100.64.1.97)  1.313 ms
4  3  100.64.0.198 (100.64.0.198)  1.443 ms 100.64.0.62 (100.64.0.62)  2.160 ms 100.64.0.60
5 (100.64.0.60)  2.116 ms
6 10  66.249.94.214 (66.249.94.214)  6.313 ms  7.104 ms 209.85.249.34 (209.85.249.34)  5.986 ms
7 11  209.85.244.65 (209.85.244.65)  6.157 ms  6.341 ms  6.574 m.
8 .
9 <span style="color: #ff6600;">12</span>  sea09s18-in-f3.1e100.net (173.194.33.163)  6.302 ms  6.517
  ms  6.071 ms
  ubuntu@devopscube:~$
```

The above output shows the hop count (12) to reach google.com from devopscube AWS ec2 server.

d) Hostname

Hostname command is used to view the hostname of the machine and to set the hostname.

```
1 ubuntu@devopscube:~$ hostname
2 devopscube.com
3 ubuntu@devopscube:~$
```

You can use the hostname command to set a new hostname for the machine.

```
1 ubuntu@devopscube:~$ sudo hostname temp.com
2 ubuntu@devopscube:~$ hostname
```

```
3 temp.com
4 ubuntu@devopscube:~$
```

If you set the hostname using “hostname” command, when you restart the machine, the hostname will change to the name specified in the hostname file (eg: /etc/hostname). So if you want to change the hostname permanently, you can use the /etc/hosts file or relevant hostname file present on the server.

For ubuntu machines, you can change it in the **/etc/hostname** file.

For RHEL, CentOS and Fedora you can change it in the **/etc/sysconfig/network** file.

e)**Netstat(Ss)**

ss command is a replacement for netstat. You can still use netstat command on all systems. Using ss command, you can get more information than netstat command. ss command is fast because it gets all the information from the kernel userspace. Now let’s have a look at few usages of ss command.

Listing All Connections

The “ss” command will list all the TCP, UDP and Unix socket connections on your machine.

```
1 ubuntu@devopscube:~$ ss
2 Netid  State      Recv-Q Send-Q   Local Address:Port       Peer Address:Port
3 u_str  ESTAB                       * 7594                *
4 u_str  ESTAB                @/com/ubuntu/upstart 7605              *
5 u_str  ESTAB                       * 29701               *
6 u_str  ESTAB                /var/run/dbus/system_bus_socket 29702              *
7 tcp    ESTAB          400      172.31.18.184:ssh        1.22.167.31:61808
```

The output of ss command will be big so you can use ” ss | less ” command to make the output scrollable.

Filtering Out TCP, UDP And Unix Sockets

If you want to filter out TCP , UDP or UNIX socket details, use “-t” “-u” and “-x” flag with the “ss” command. It will show all the established connections to the specific ports. If you want to list both connected and listening ports using “a” with the specific flag as shown below.

```
1 ss -ta
2 ss -ua
3 ss -xa
```

List All Listening Ports

To list all the listening ports, use “-l” flag with ss command. To list specific TCP, UDP or UNIX socket, use “-t”, “-u” and “-x” flag with “-l” as shown below.

```
1 ubuntu@devopscube:~$ ss -lt
2 State      Recv-Q Send-Q     Local Address:Port         Peer Address:Port
3 LISTEN     128                  *:ssh                   *:*
4 LISTEN     50                 :::http-alt             :::*
5 LISTEN     50                 :::55857                :::*
6 LISTEN     128                 :::ssh                 :::*
7 LISTEN     50                 :::53285                :::*
```

```
8 ubuntu@devopscube:~$
```

f) Route

"route" command is used to get the details of route table for your system and to manipulate it. Let us look at few examples for the route command.

Listing All Routes

Execute the "route" command without any arguments to list all the existing routes in your system or server.

```
1 ubuntu@devopscube:~$ route
2 Kernel IP routing table
3 Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
4 default         ip-172-31-16-1. 0.0.0.0         UG                    eth0
5 172.17.0.0      *               255.255.0.0     U                     docker0
6 172.31.16.0     *               255.255.240.0   U                     eth0
7 ubuntu@devopscube:~$
```

If you want to get the full output in numerical form without any hostname, you can use "-n" flag with the route command.

```
ubuntu@devopscube:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.31.16.1     0.0.0.0         UG                    eth0
172.17.0.0      0.0.0.0         255.255.0.0     U                     docker0
172.31.16.0     0.0.0.0         255.255.240.0   U                     eth0
ubuntu@devopscube:~$
```

g) Whois:

whois COMMAND:

whois command lists the information about the domain owner of the given domain.

SYNTAX :
whois [option] query

OPTIONS:

-h Host which holds the identification information in its database.
-p connect to the specified port.

EXAMPLE:

1. whoishscripts.com

Output:
The above command will produce the following output.

[Querying whois.internic.net] [Redirected to whois.PublicDomainRegistry.com] [Querying whois.PublicDomainRegistry.com] [whois.PublicDomainRegistry.com] Registration Service Provided By: HIOX INDIA Contact: +91.4226547769 Domain Name: HSCRIPTS.COM
Registrant:
HIOX INDIA
Rajesh Kumar
(support@hioxindia.com)
32, North Street, Krishnapuram, Singanallur
Coimbatore
tamil nadu,641005
IN
Tel. +91.04225547769 Creation Date: 06-Oct-2004 Expiration Date: 06-Oct-2008 Domain servers in listed order:
ns1.hscripts.com
ns2.hscripts.com

h) NSlookup:

nslookup, which stands for "name server lookup", is a useful tool for finding out information about a named domain.

By default, **nslookup** will translate a domain name to an IP address (or vice versa). For instance, to find out what the IP address of **microsoft.com** is, you could run the command:

```
nslookup microsoft.com
```

...and you would receive a response like this:

```
Server:     8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
Name:    microsoft.com
Address: 134.170.185.46
Name:    microsoft.com
Address: 134.170.188.221
```

Here, **8.8.8.8** is the address of our system's Domain Name Server. This is the server our system is configured to use to translate domain names into IP addresses. "**#53**" indicates that we are communicating with it on port 53, which is the standard port number domain name servers use to accept queries.We can also perform the above operation in reverse by providing the IP address rather than the domain name. For instance, the command:

```
nslookup 134.170.185.46
```

...will return information resembling the following:

```
Server:     8.8.8.8
Address:    8.8.8.8#53
```

i)Arp:

arp manipulates or displays the kernel's IPv4 network neighbour cache. It can add entries to the table, delete one, or display the current content.

ARP stands for **Address Resolution Protocol**, which is used to find the address of a network neighbor for a given IPv4 address.

-H type, --hw-Type type	When setting or reading the ARP cache, this optional parameter tells arp which class of entries it should check for. The default value of this parameter is ether (i.e. hardware code 0x01 for IEEE 802.3 10Mbps Ethernet). Other values might include network technologies such as ARCnet (arcnet) , PROnet (pronet), AX.25 (ax25) and NET/ROM (netrom).
-a [hostname]	Use alternate BSD-style output format (with no fixed columns).
-D, --use-device	Instead of a hw_addr, the given argument is the name of an interface. arp will use the MAC address of that interface for the table entry. This is usually the best option to set up a proxy ARP entry to yourself.
-i If, --device If	Select an interface. When dumping the ARP cache only entries matching the specified interface will be printed. When setting a permanent or temp ARP entry this interface will be associated with the entry; if this option is not used, the kernel will guess based on the routing table. For pub entries the specified interface is the interface on which ARP requests will be answered. • **NOTE:** This has to be different from the interface to which the IP datagrams will be routed.
-f filename, --file filename	Similar to the -s option, only this time the address info is taken from file filename.This can be used if ARP entries for a lot of hosts have to be set up.The name of the data file is very often /etc/ethers, but tis not official. If no filename is specified /etc/ethers is used as default. • In all places where a hostname is expected, one can also enter an IP address in dotted-decimal notation

Example-1:

arp

output:

Address	HWtype	HWaddress	Flags Mask	Iface
192.168.0.103	ether	48:e2:44:d5:7a:97	C	eth0
192.168.0.1	ether	c8:3a:35:49:77:48	C	eth0

Example-2:

To delete a ARP table entry. Root privilege is required to do this.

sudo arp -d 192.168.0.1

output:

Address HWtype HWaddress Flags Mask Iface
192.168.0.103 ether 48:e2:44:d5:7a:97 C eth0
192.168.0.1 (incomplete) eth0
(In this case the arp entry is deleted which was avaiable in the earlier section, and is shown as incomplete)

Example-3:

Answer ARP request on eth0 with the MAC address for eth1

```
#arp -i eth0 -Ds 192.168.0.104 eth1 pub

In this case 192.168.0.104 is the ip address  associated with eth0 and we are
configuring it to answer ARP request  on eth0 with the MAC address for eth1.
```

Example-4:

By default the arp command will show the hostname of the items within the ARP cache but you can force it to display IP addresses using the following switch:

```
#arp -n

output:
Address              HWtype   HWaddress               Flags Mask    Iface
192.168.0.103    ether     48:e2:44:d5:7a:97     C                  eth0
192.168.0.1        ether     c8:3a:35:49:77:48     C                  eth0
```

Example-5:

Alternatively you might wish to use the following switch which will display the output in a different way:

```
#arp -a
Output:
? (192.168.0.103) at 48:e2:44:d5:7a:97 [ether] on eth0
? (192.168.0.1) at c8:3a:35:49:77:48 [ether] on eth0
```

j) Finger:

The **finger** displays information about the system users.

Options are:

Tag	Description

-s	**Finger** displays the user's login name, real name, terminal name and write status (as a ''*'' after the terminal name if write permission is denied), idle time, login time, office location and office phone number. Login time is displayed as month, day, hours and minutes, unless more than six months ago, in which case the year is displayed rather than the hours and minutes. Unknown devices as well as nonexistent idle and login times are displayed as single asterisks.
-l	Produces a multi-line format displaying all of the information described for the **-s** option as well as the user's home directory, home phone number, login shell, mail status, and the contents of the files "`.plan`", "`.project`", "`.pgpkey`" and "`.forward`" from the user's home directory. Phone numbers specified as eleven digits are printed as ''+N-NNN-NNN-NNNN''. Numbers specified as ten or seven digits are printed as the appropriate subset of that string. Numbers specified as five digits are printed as ''xN-NNNN''. Numbers specified as four digits are printed as ''xNNNN''. If write permission is denied to the device, the phrase ''(messages off)'' is appended to the line containing the device name. One entry per user is displayed with the **-l** option; if a user is logged on multiple times, terminal information is repeated once per login. Mail status is shown as ''No Mail.'' if there is no mail at all, ''Mail last read DDD MMM ## HH:MM YYYY (TZ)'' if the person has looked at their mailbox since new mail arriving, or ''New mail received ...'', '' Unread since ...'' if they have new mail.
-p	Prevents the **-l** option of **finger** from displaying the contents of the "`.plan`", "`.project`" and "`.pgpkey`" files.
-m	Prevent matching of *user* names. *User* is usually a login name; however, matching will also be done on the users' real names, unless the **-m** option is supplied. All name matching performed by **finger** is case insensitive.

If no options are specified, **finger** defaults to the **-l** style output if operands are provided, otherwise to the **-s** style. Note that some fields may be missing, in either format, if information is not available for them.

If no arguments are specified, **finger** will print an entry for each user currently logged into the system.

Finger may be used to look up users on a remote machine. The format is to specify a *user* as "user@host", or "@host", where the default output format for the former is the **-l** style, and the

default output format for the latter is the **-s** style. The **-l** option is the only option that may be passed to a remote machine.

If standard output is a socket, **finger** will emit a carriage return (^M) before every linefeed (^J). This is for processing remote finger requests when invoked by **fingerd**(8).

k) Port Scan / nmap:

Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. Nmap is used for exploring networks, perform security scans, network audit and finding open ports on remote machine. It scans for Live hosts, Operating systems, packet filters and open ports running on remote hosts.

Most of the today's Linux distributions like **Red Hat**, **CentOS**, **Fedoro**, **Debian** and **Ubuntu** have included **Nmap** in their default package management repositories called Yum and APT. The both tools are used to install and manage software packages and updates. To install **Nmap** on distribution specific use the following command.

```
# yum install nmap             [on Red Hat based systems]
$ sudo apt-get install nmap    [on Debian based systems]
```

Scan using IP Address

```
[root@server1 ~]# nmap 192.168.0.101
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2013-11-18 11:04 EST
Interesting ports on server2.tecmint.com (192.168.0.101):
Not shown: 1674 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
111/tcp  open  rpcbind
958/tcp  open  unknown
3306/tcp open  mysql
8888/tcp open  sun-answerbook
MAC Address: 08:00:27:D9:8E:D7 (Cadmus Computer Systems)
Nmap finished: 1 IP address (1 host up) scanned in 0.465 seconds
You have new mail in /var/spool/mail/root
```

Conclusion: We have implemented network configuration & troubleshooting commands.