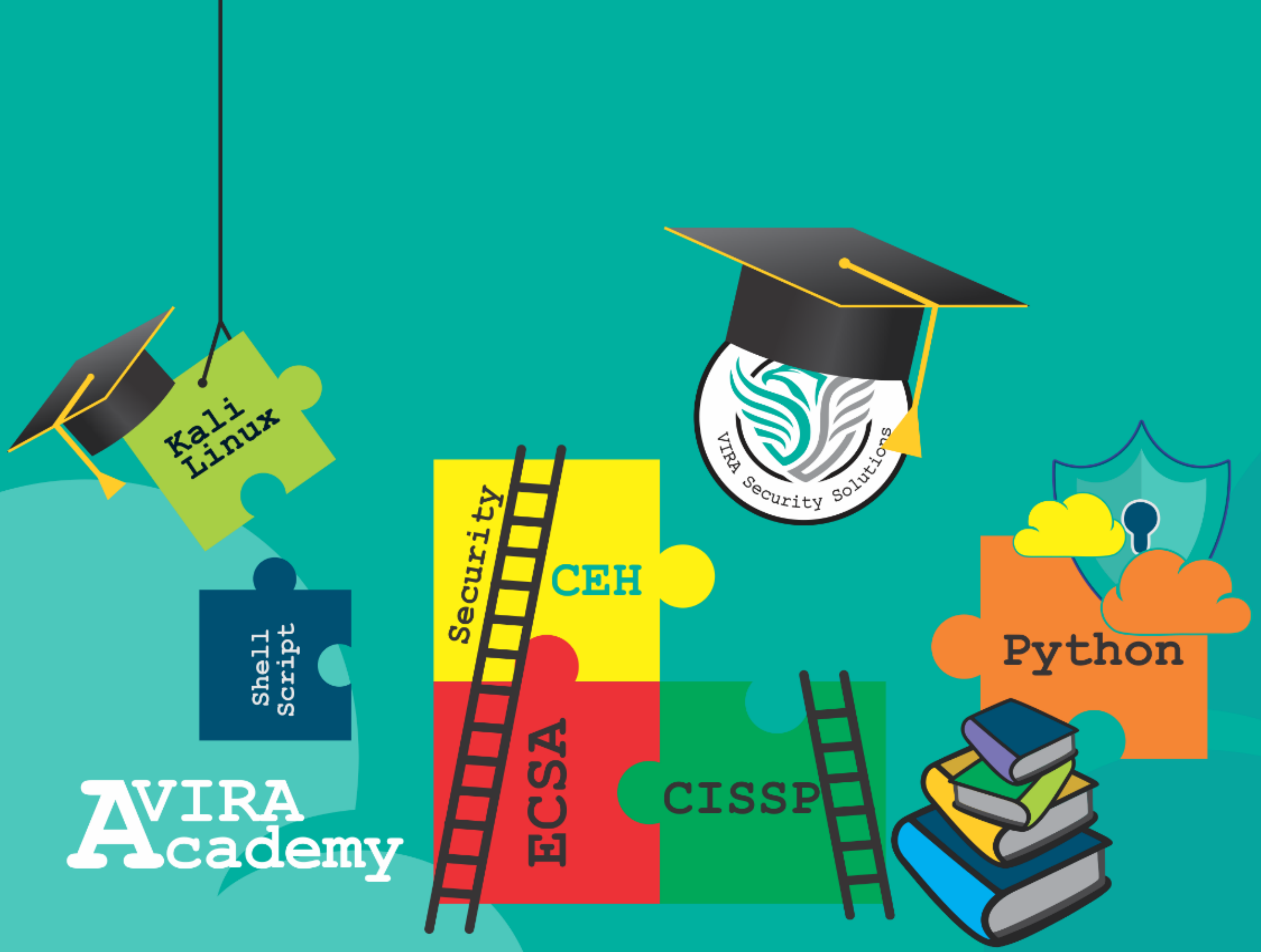
VIRA Security Solutions

راهکارهای امن ویرا

شماره ثبت: ۵۴۰۱۳۹

VIRA Security Solutions


Kali Linux
Shell Script
Security
CEH
ECSA
CISSP
Python
VIRA Security Solutions
AVIRA Academy

Linux

Part4

Mohammad Reza Gerami
Mrgerami@aut.ac.ir
gerami@virasecsolutions.com

April 10 2020

Regular Expressions

Regular Expressions

Introduction

Regular expressions are a very powerful tool in Linux. They can be used with a variety of programs like bash, vi, rename, grep, sed, and more.

Regular Expressions

regex versions

There are three different versions of regular expression syntax:

BRE: Basic Regular Expressions
ERE: Extended Regular Expressions
PRCE: Perl Regular Expressions

Regular Expressions

Depending on the tool being used, one or more of these syntaxes can be used.

For example, the grep tool has the -E option to force a string to be read as ERE while -G forces BRE and -P forces PRCE.

Note that grep als has -F to force the string to be read literally.

Find files of type file (not directory, pipe or etc.) that end in .conf.

The sed tool also has options to choose a regex syntax.

Read the manual of the tools you use!

Regular Expressions

Word brackets The “\<” matches the beginning of a word (a place where a nonletter precedes a letter). Analogously, “\>” matches the end of a word (where
a letter is followed by a non-letter).

Grouping Parentheses (“(…)”) allow for the repetition of concatenations of regular expressions: “a(bc)*” matches a “a” followed by arbitrarily many repetitions of “bc”.

Alternative With the vertical bar (“|”) you can select between several regular expressions. The expression “motor (bike|cycle|boat)” matches “motor bike”,
“motor cycle”, and “motor boat” but nothing else.

Regular Expressions

Optional Expression The question mark (“?”) makes the preceding regular expression optional, i. e., it must occur either once or not at all. “ferry(man)?” matches either “ferry” or “ferryman”.

At-Least-Once Repetition The plus sign (“+”) corresponds to the repetition operator “*”, except that the preceding regular expression must occur at least once.

Given Number of Repetitions You can specify a minimum and maximum number of repetitions in braces: “ab{2,4}” matches “abb”, “abbb”, and “abbbb”, but not “ab” or “abbbbb”. You may omit the minimum as well as the maximum number; if there is no minimum number, 0 is assumed, if there is no maximum number, “infinity” is assumed.

Regular Expressions

Back-Reference With an expression like “\\n you may call for a repetition of that part of the input that matched the parenthetical expression no. n in the regular expression. “(ab)\\1”, for example, matches “abab”, and if, when processing “(ab*a)x\1”, the parentheses matched abba, then the whole expression matches abbaxabba (and nothing else). More detail is available in the documentation of GNU grep.

Non-Greedy Matching The “*”, “+”, and “?” operators are usually “greedy”, i. e.,
they try to match as much of the input as possible: “^a.*a” applied to the input string “abacada” matches “abacada”, not “aba” or “abaca”. However, there are corresponding “non-greedy” versions “*?”, “+?”, and “??” which try to match as little of the input as possible. In our example, “^a.*?a” would match “aba”. The braces operator may also offer a non-greedy version.

Regular Expressions

grep Global Regular Expression Print

print lines matching a pattern

grep is a popular Linux tool to search for lines that match a certain pattern. Below are some examples of the simplest regular expressions.

This is the contents of the text file. This file contains four lines (or four newline characters).

Regular Expressions

grep Global Regular Expression Print

print lines matching a pattern

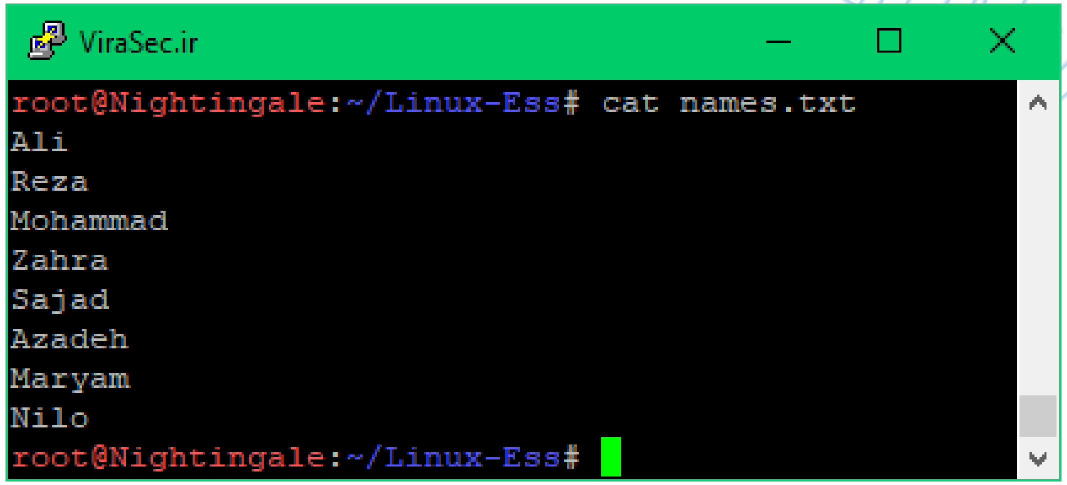
Option		Result
`-c`	(*count*)	Outputs just the number of matching lines
`-i`	(*ignore*)	Uppercase and lowercase letters are equivalent
`-l`	(*list*)	Outputs just the names of matching files, no actual matches
`-n`	(*number*)	Includes line numbers of matching lines in the output
`-r`	(*recursive*)	Searches files in subdirectories as well
`-v`	(*invert*)	Outputs only lines that do *not* match the regular expression

Regular Expressions

grep

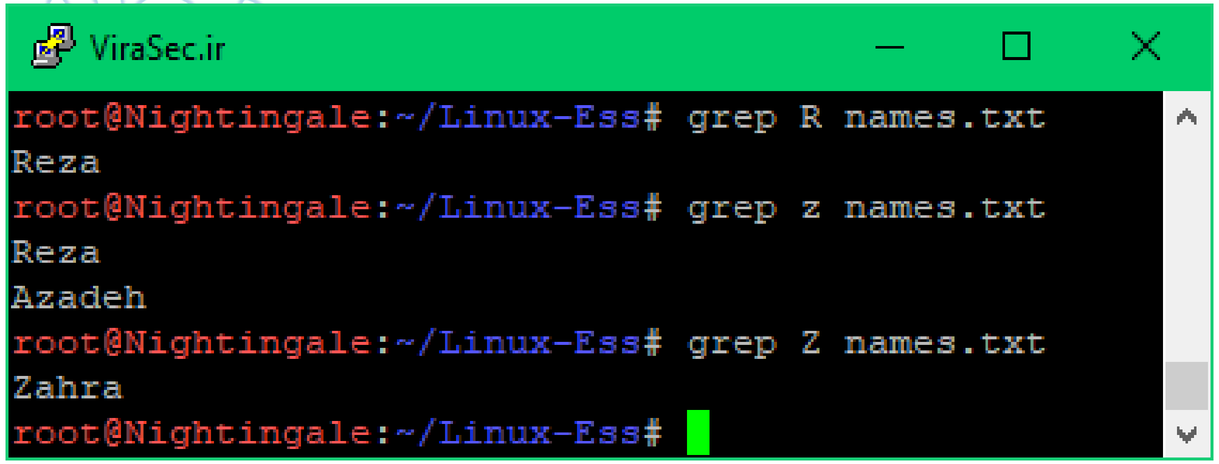
print lines matching a pattern

When grepping for a single character, only the lines containing that character are returned.



ViraSec.ir

```
root@Nightingale:~/Linux-Ess# cat names.txt
Ali
Reza
Mohammad
Zahra
Sajad
Azadeh
Maryam
Nilo
root@Nightingale:~/Linux-Ess#
```



ViraSec.ir

```
root@Nightingale:~/Linux-Ess# grep R names.txt
Reza
root@Nightingale:~/Linux-Ess# grep z names.txt
Reza
Azadeh
root@Nightingale:~/Linux-Ess# grep Z names.txt
Zahra
root@Nightingale:~/Linux-Ess#
```

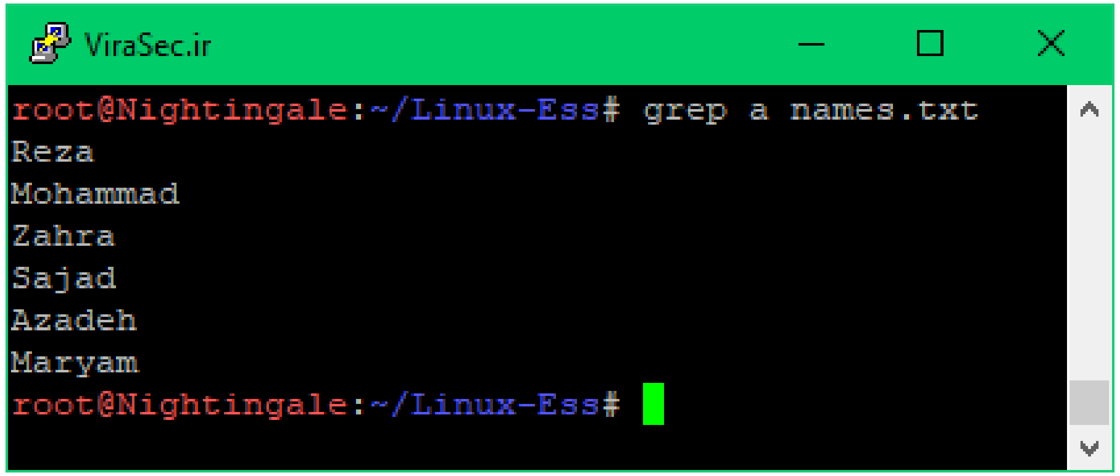
Regular Expressions

grep

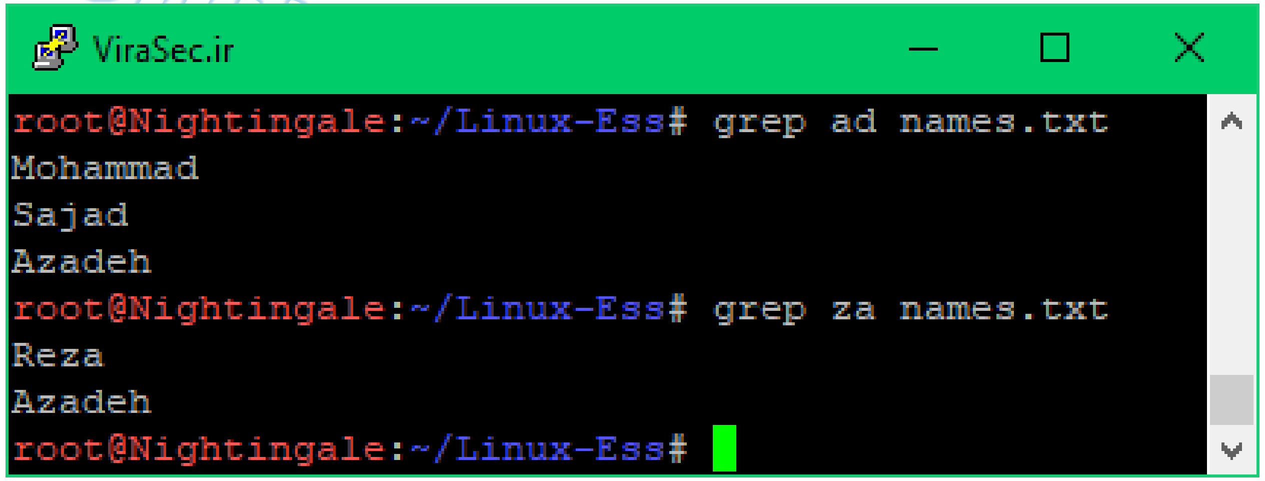
print lines matching a pattern

concatenating characters

Two concatenated characters will have to be concatenated in the same way to have a match.



```
root@Nightingale:~/Linux-Ess# grep a names.txt
Reza
Mohammad
Zahra
Sajad
Azadeh
Maryam
root@Nightingale:~/Linux-Ess#
```



```
root@Nightingale:~/Linux-Ess# grep ad names.txt
Mohammad
Sajad
Azadeh
root@Nightingale:~/Linux-Ess# grep za names.txt
Reza
Azadeh
root@Nightingale:~/Linux-Ess#
```

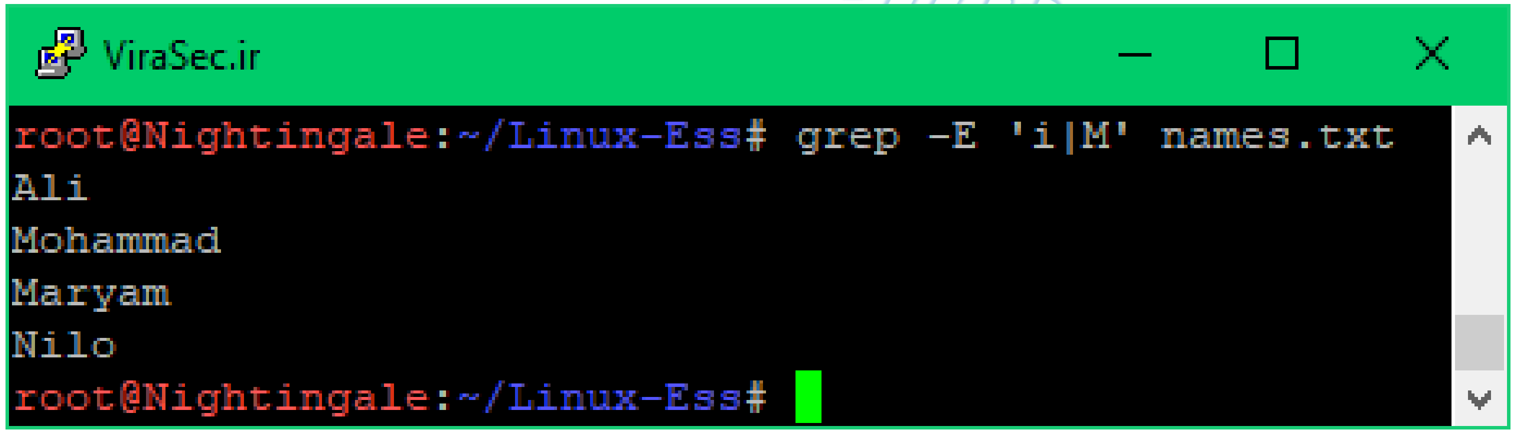
Regular Expressions

grep

print lines matching a pattern

one or the other

PRCE and ERE both use the pipe symbol to signify OR. In this example we grep for lines containing the letter i or the letter M.



```
root@Nightingale:~/Linux-Ess# grep -E 'i|M' names.txt
Ali
Mohammad
Maryam
Nilo
root@Nightingale:~/Linux-Ess#
```

Note that we use the -E switch of grep to force interpretion of our string as an ERE

Regular Expressions

Searching Files

cat report.c	{prints file on stdout, no pauses}
cat -v -e -t dump	{show non-printing characters too}
cat >newfile	{reads from stdin, writes to 'newfile'}
cat rpt1.c inp.c test.s >newfile	{combine 3 files into 1}
more report.c	{space for next page, q to quit}
ps -a | more	{page through the full output of ps}
grep vira *.txt	{search *.txt files for 'vira'}

Regular Expressions

Searching Files Using UNIX grep

% grep BOB tmpfile	{search 'tmpfile' for 'BOB' anywhere in a line}
% grep -i -w blkptr *	{search files in CWD for word blkptr, any case}
% grep run[-]time *.txt	{find 'run time' or 'run-time' in all txt files}
% who | grep root	{pipe who to grep, look for root}

Regular Expressions

Understanding Regular Expressions

^ (Caret)	=	match expression at the start of a line, as in ^A.
$ (Question)	=	match expression at the end of a line, as in A$.
\ (Back Slash)	=	turn off the special meaning of the next character, as in \^.
[ ] (Brackets)	=	match any one of the enclosed characters, as in [aeiou]. Use Hyphen "-" for a range, as in [0-9].
[^]	=	match any one character except those enclosed in [ ], as in [^0-9].
. (Period)	=	match a single character of any value, except end of line.
* (Asterisk)	=	match zero or more of the preceding character or expression.
\{x,y\}	=	match x to y occurrences of the preceding.
\{x\}	=	match exactly x occurrences of the preceding.
\{x,\}	=	match x or more occurrences of the preceding.

Regular Expressions

Understanding Regular Expressions

Since you usually type regular expressions within shell commands, it is good practice to enclose the regular expression in single quotes (') to stop the shell from expanding it before passing the argument to your search tool. Here are some examples using grep:

grep vira files	{search files for lines with 'vira'}
grep '^vira' files	{'vira' at the start of a line}
grep 'vira$' files	{'vira' at the end of a line}
grep '^vira$' files	{lines containing only 'vira'}
grep '^\^s' files	{lines starting with '^s', "\" escapes the ^}
grep '[Ss]mug' files	{search for 'vira' or 'vira'}
grep 'B[oO][bB]' files	{search for BOB, Bob, BOb or BoB }
grep '^$' files	{search for blank lines}
grep '[0-9][0-9]' file	{search for pairs of numeric digits}

Regular Expressions

Understanding Regular Expressions

Back Slash "\" is used to escape the next symbol, for example, turn off the special meaning that it has. To look for a Caret "^" at the start of a line, the expression is ^\^. Period "." matches any single character. So b.b will match "bob", "bib", "b-b", etc. Asterisk "*" does not mean the same thing in regular expressions as in wildcarding; it is a modifier that applies to the preceding single character, or expression such as [0-9]. An asterisk matches zero or more of what precedes it. Thus [A-Z]* matches any number of upper-case letters, including none, while [A-Z][A-Z]* matches one or more upper-case letters.

The vi editor uses \< \> to match characters at the beginning and/or end of a word boundary. A word boundary is either the edge of the line or any character except a letter, digit or underscore "_". To look for if, but skip stiff, the expression is \<if\>. For the same logic in grep, invoke it with the -w option. And remember that regular expressions are case-sensitive. If you don't care about the case, the expression to match "if" would be [Ii][Ff], where the characters in square brackets define a character set from which the pattern must match one character. Alternatively, you could also invoke grep with the -i option to ignore case.

Regular Expressions

Here are a few more examples of **grep** to show you what can be done:

grep '^From: ' /usr/mail/$USER	{list your mail}
grep '[a-zA-Z]'	{any line with at least one letter}
grep '[^a-zA-Z0-9]	{anything not a letter or number}
grep '[0-9]\{3\}-[0-9]\{4\}'	{999-9999, like phone numbers}
grep '^.$'	{lines with exactly one character}
grep '"vira"'	{'vira' within double quotes}
grep '"*vira"*'	{'vira', with or without quotes}
grep '^\.'	{any line that starts with a Period "."}
grep '^\.[a-z][a-z]'	{line start with "." and 2 lc letters}

Regular Expressions

find all lines containing exactly the word

```
grep \<Ali\> name.txt
```

find all lines *beginning* with "Ali"

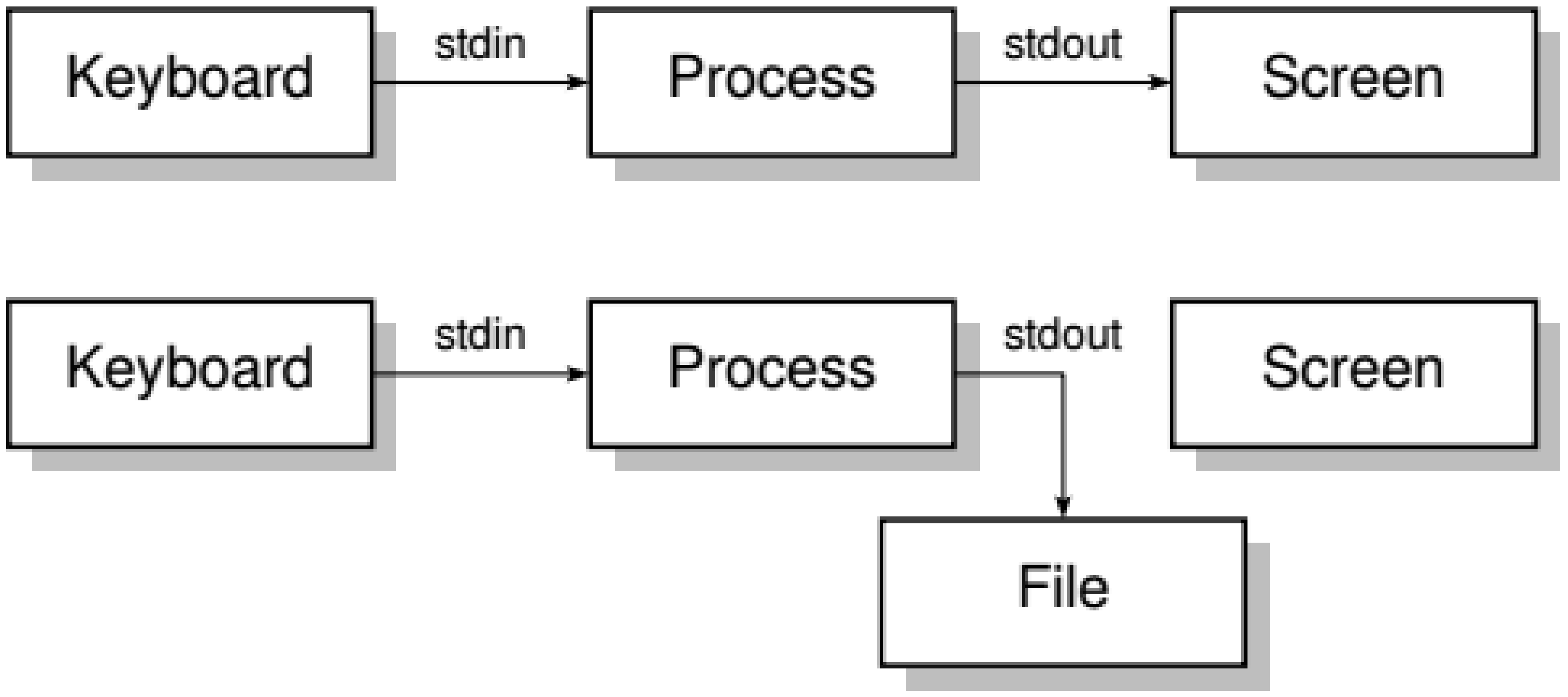
```
grep ^Ali name.txt
```

All words containing three or more "a"

```
grep -n 'a.*a.*a' /usr/share/dict/words
```

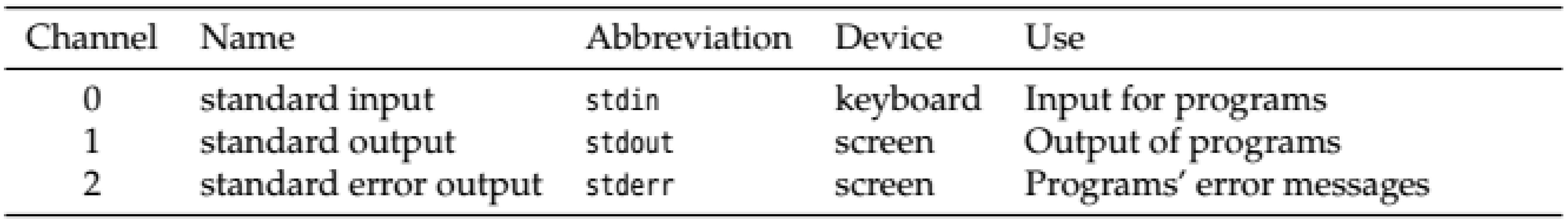
Standard I/O and Filter Commands

Standard channels on Linux



Standard I/O and Filter Commands

Standard channels on Linux



Channel	Name	Abbreviation	Device	Use
0	standard input	`stdin`	keyboard	Input for programs
1	standard output	`stdout`	screen	Output of programs
2	standard error output	`stderr`	screen	Programs' error messages

How to configure a static IP address on CentOS 7 / RHEL 7

On CentOS 7 or RHEL 7 one need to use the Network Manager daemon.
It attempts to make networking configuration and operation as painless and automatic as possible by managing the primary network connection and other network interfaces, like Ethernet, WiFi, and Mobile Broadband devices.
In this quick tutorial you will learn about configuring a network interface using ifcfg files located in /etc/sysconfig/network-scripts/ directory in a CentOS 7 and RHEL 7:

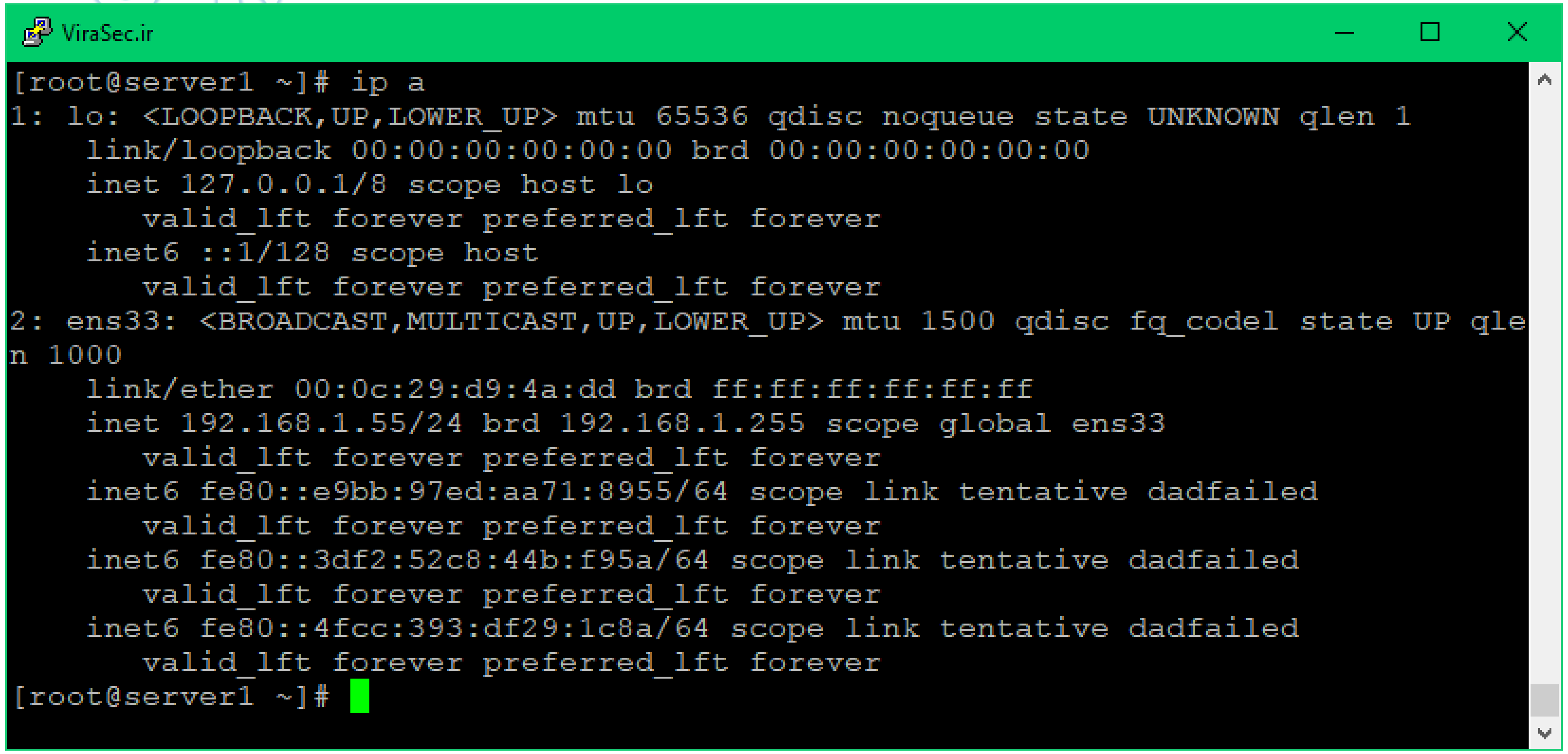
Create a file named /etc/sysconfig/network-scripts/ifcfg-eth0 as follows:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=192.168.1.203
```

Restart network service: systemctl restart network

How do I list network interfaces?

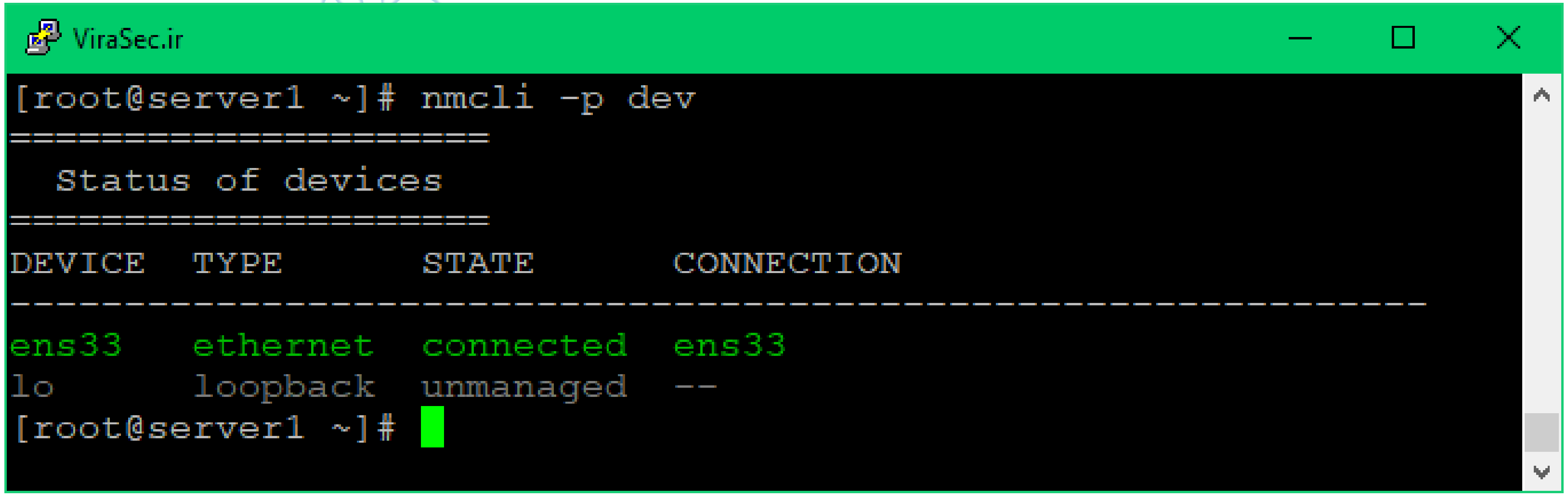
Type the following command:
# ip a



```
[root@server1 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP qle
n 1000
    link/ether 00:0c:29:d9:4a:dd brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.55/24 brd 192.168.1.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::e9bb:97ed:aa71:8955/64 scope link tentative dadfailed
       valid_lft forever preferred_lft forever
    inet6 fe80::3df2:52c8:44b:f95a/64 scope link tentative dadfailed
       valid_lft forever preferred_lft forever
    inet6 fe80::4fcc:393:df29:1c8a/64 scope link tentative dadfailed
       valid_lft forever preferred_lft forever
[root@server1 ~]#
```

How do I list network interfaces?

Or use the following command:
# nmcli -p dev



```
[root@server1 ~]# nmcli -p dev
=====================
  Status of devices
=====================
DEVICE  TYPE      STATE      CONNECTION
--------------------------------------------------------------
ens33   ethernet  connected  ens33
lo      loopback  unmanaged  --
[root@server1 ~]#
```

How do I list network interfaces?

Here is a typical DHCP configration for eth0 (stored in /etc/sysconfig/network-scripts/ifcfg-eth0 file):

```
DEVICE="eth0"
ONBOOT=yes
NETBOOT=yes
UUID="41171a6f-bce1-44de-8a6e-cf5e782f8bd6"
IPV6INIT=yes
BOOTPROTO=dhcp
HWADDR="00:08:a2:0a:ba:b8"
TYPE=Ethernet
NAME="eth0"
```

How do I configure an eth0 interface with static network settings (method # 1)?

Here is a typical DHCP configration for eth0
(stored in /etc/sysconfig/network-scripts/ifcfg-eth0 file):

To configure an eth0 interface with static network settings using ifcfg files, edit or create a file with name ifcfg-eth0 in the /etc/sysconfig/network-scripts/ directory.

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

Update/edit as follows for static IP configuration:

```
HWADDR=00:08:A2:0A:BA:B8
TYPE=Ethernet
BOOTPROTO=none
# Server IP #
IPADDR=192.168.2.203
# Subnet #
PREFIX=24
# Set default gateway IP #
GATEWAY=192.168.1.254
# Set dns servers #
DNS1=192.168.1.254
DNS2=8.8.8.8
DNS3=8.8.4.4
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
# Disable ipv6 #
IPV6INIT=no
NAME=eth0
# This is system specific and can be created using 'uuidgen eth0' command #
UUID=41171a6f-bce1-44de-8a6e-cf5e782f8bd6
DEVICE=eth0
ONBOOT=yes
```

Save and close the file. You do not need to specify the network or broadcast address as this is calculated automatically by the system. To restart networking service, enter:

```
# systemctl restart network
```

Verification

Verify new IP settings:

```
# ip a s ens33
```

Verify new routing settings:

```
# ip r
```

Verify DNS servers settings:

```
# cat /etc/resolv.conf
```

Verify the internet connectivity:

```
# ping -c 3 virasecsolutions.com
# ping -c 4 google.com
```

Network config

For short-term experiments you can also use the ifconfig command. Something like:

```
# ifconfig eth0 192.168.178.111 netmask 255.255.255.0
```

assigns the IP address 192.168.178.111 to the network interface eth0 (Ethernet). The local network is 192.168.178.0/24—the 255.255.255.0 is a roundabout method of writing down the subnet mask (24). The default gateway (e. g, 192.168.178.1) is configured using the route command:

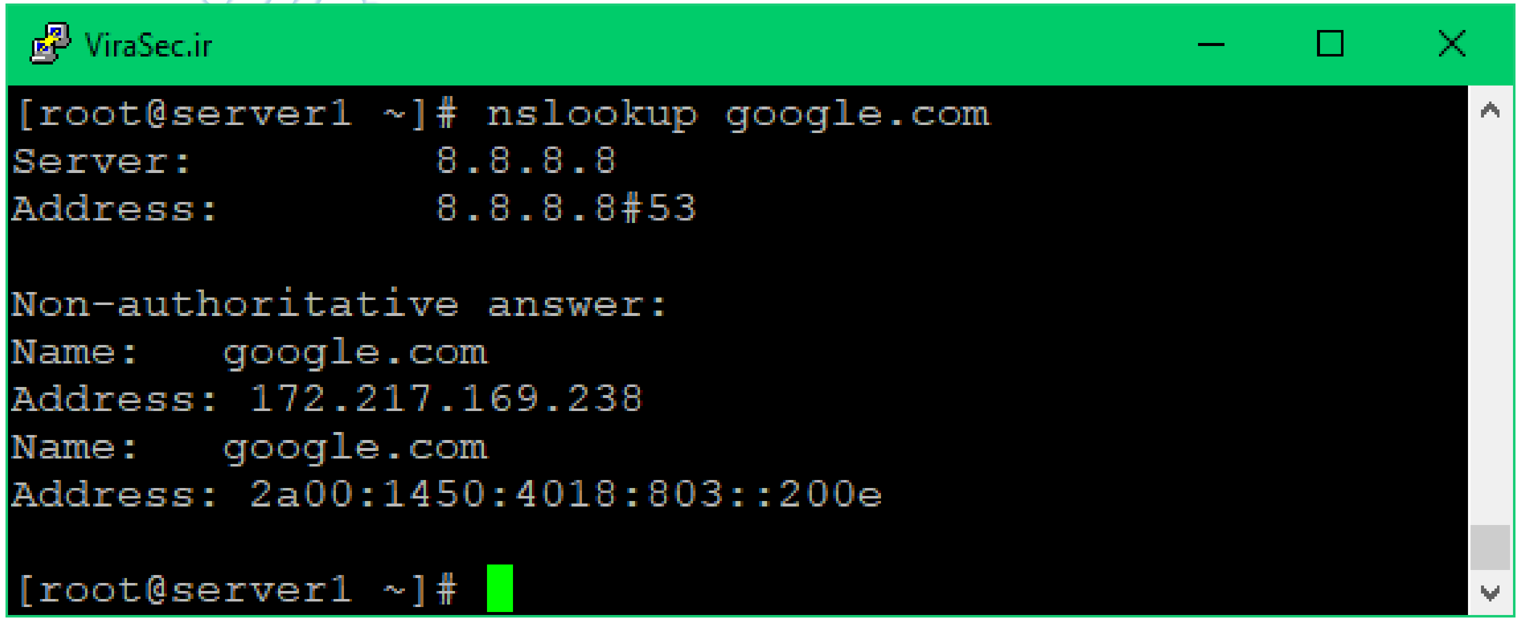
```
# route add -net default gw 192.168.178.1
```

These settings only persist until the next reboot!

DNS

The addresses of DNS servers are usually stored in the /etc/resolv.conf file, which might look approximately like

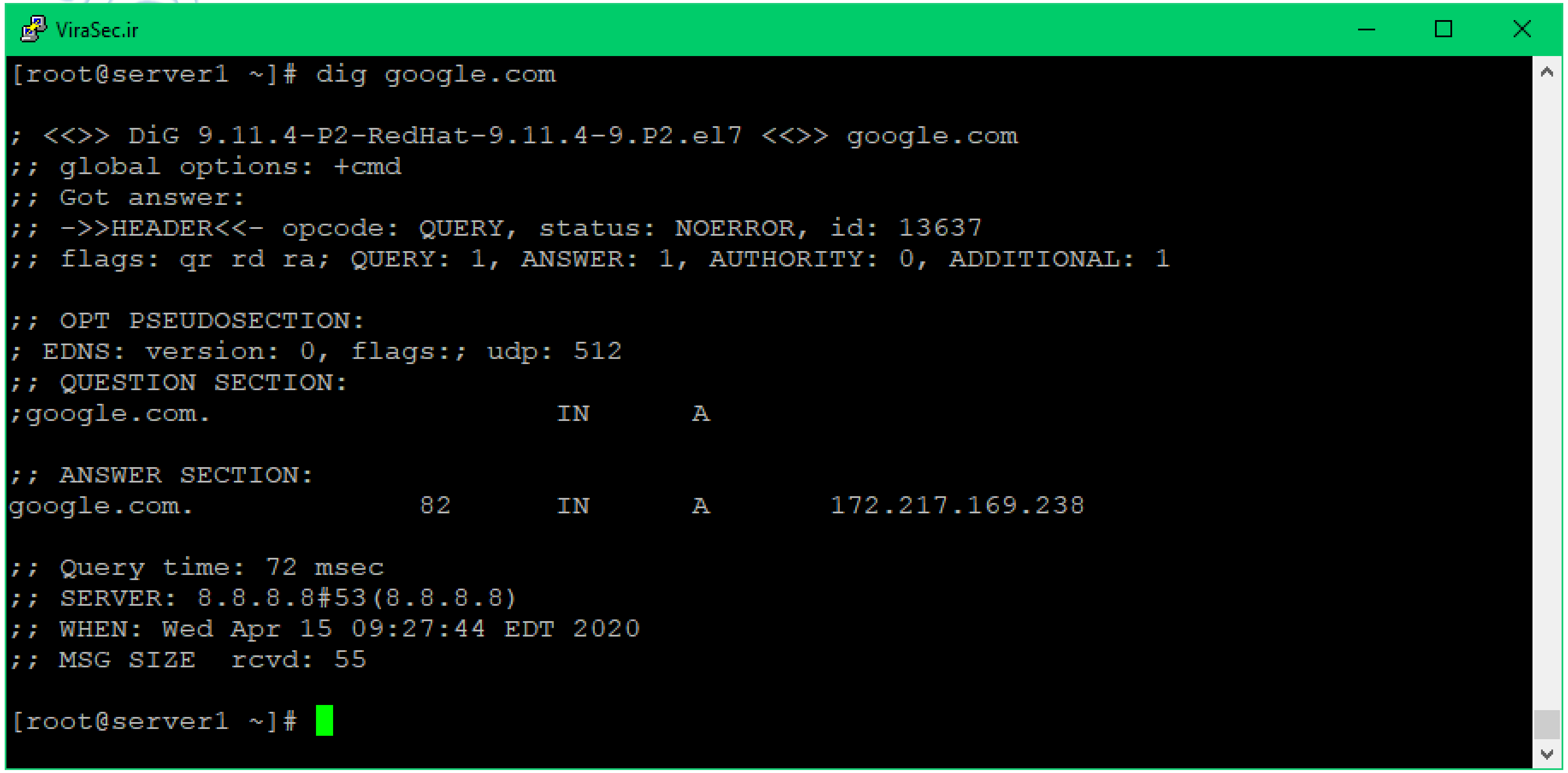
```
# /etc/resolv.conf
nameserver 8.8.8.8
nameserver 4.2.2.4
```



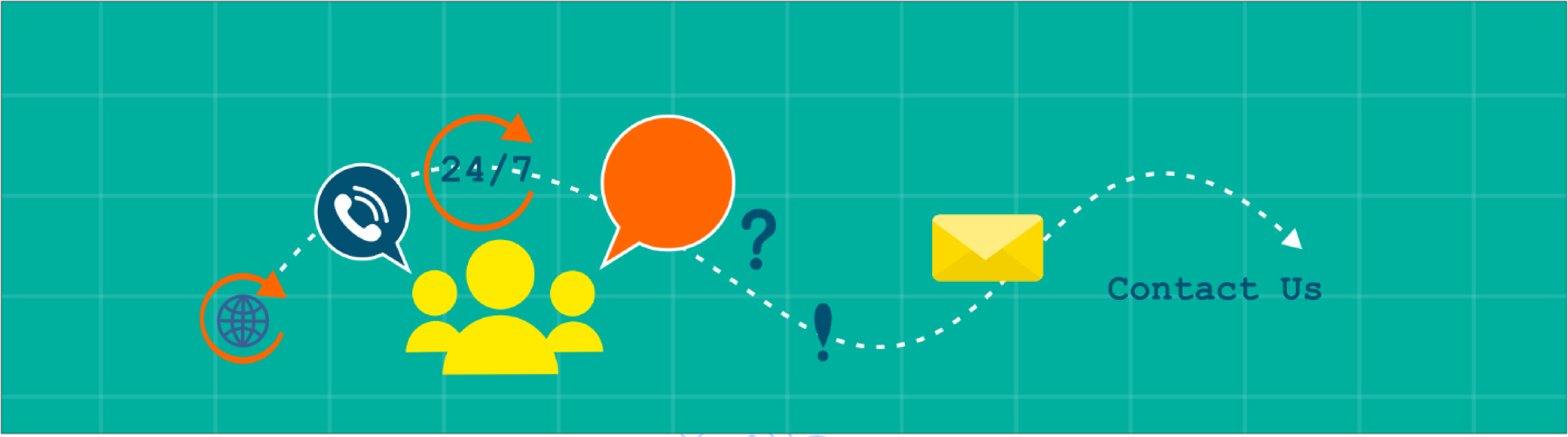
```
[root@server1 ~]# nslookup google.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.169.238
Name:   google.com
Address: 2a00:1450:4018:803::200e

[root@server1 ~]#
```

Verification

dig The dig command is used to test DNS name resolution. Unless you specify otherwise, it tries to find an IP address corresponding to a name given on the command line:



```
[root@server1 ~]# dig google.com

; <<>> DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13637
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             82      IN      A       172.217.169.238

;; Query time: 72 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Apr 15 09:27:44 EDT 2020
;; MSG SIZE  rcvd: 55

[root@server1 ~]#
```



Visiting Address: Unit 20, Floor 4, No 53 Vafa Manesh Ave

Heravi, Pasdaran Ave, TEHRAN-IRAN

Post Code:1668838803

Tel No: 0098 21 2298 1027-09125792641

Email: info@ virasecsolutions.com

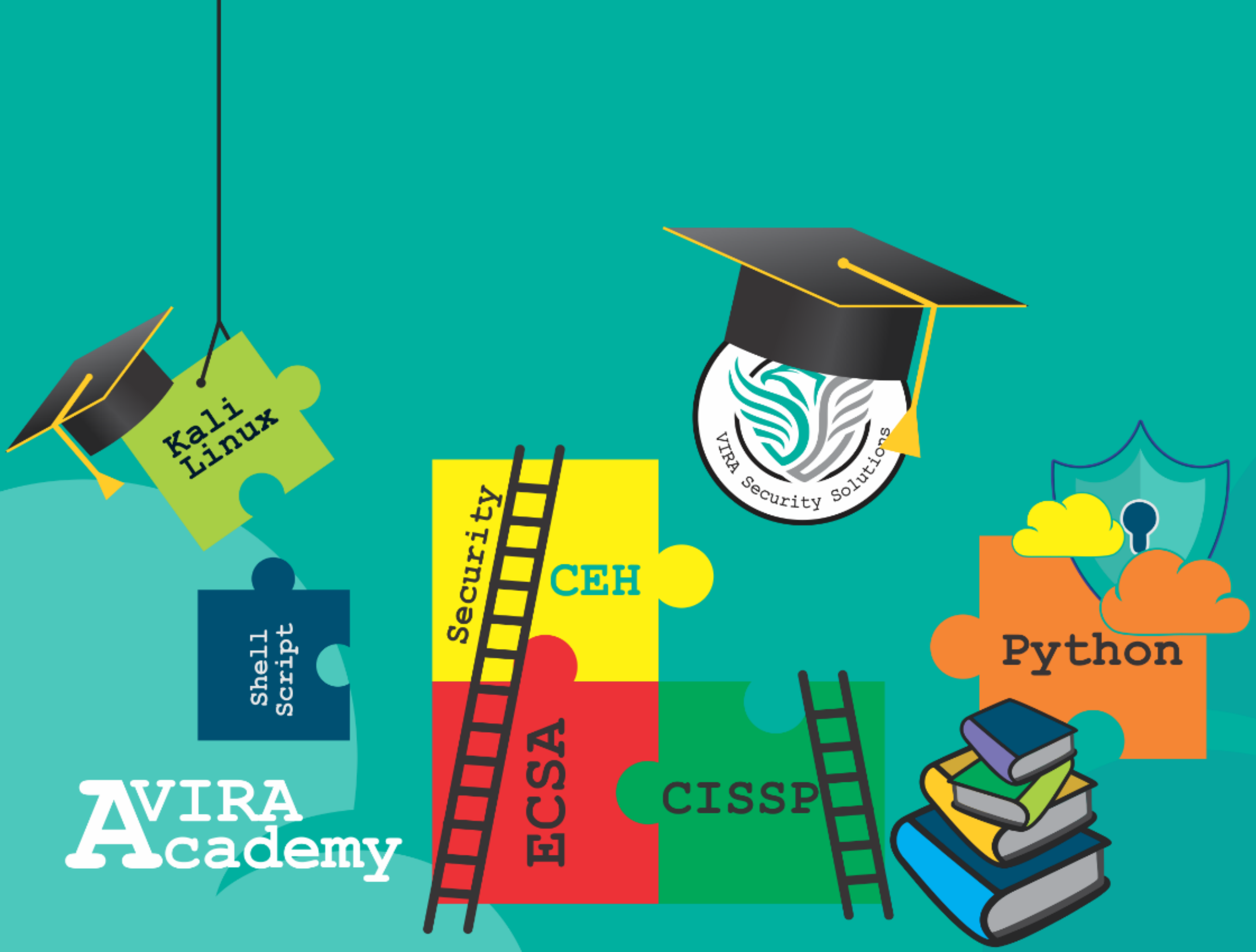
Website: www.virasecsolutions.com

آدرس: تهران، پاسداران، هروی، خیابان وفامنش، پلاک ۵۳

طبقه چهارم، واحد ۲۰

کد پستی: ۱۶۶۸۸۳۸۸۰۳

شماره تماس: ۰۲۱۲۲۹۸۱۰۲۷-۰۹۱۲۵۷۹۲۶۴۱


Kali Linux
Shell Script
Security
CEH
ECSA
CISSP
Python
VIRA Security Solutions
VIRA Academy