Many times, we use terminal to remote access into our unix server to run java server process or java services. However we cant type anything After we executed java program like following

debian:/QueryTestServer# java QueryTestServer

In addition, the java program will automatically killed after we terminated our remote shell prompt. This is not what we want. Is there any command that can run a java program in Unix background environment?

Yes it is. In order to **run a java program in unix background environment**, we need to append “**&**” to the end of our command.

debian:/QueryTestServer# java QueryTestServer &

Done, the java program will executed in Unix background environment until we explicitly kill it.

**Using Java via Command Line**

|  |
| --- |
| For some classes, you will need to test your Java code on the lab computers. You can [log in remotely](http://sjengle.cs.usfca.edu/resources/logging-in-remotely), and then run your Java code via the command-line. You can find some basic information about the java command-line tool in the [JDK documentation](http://docs.oracle.com/javase/6/docs/technotes/tools/index.html#basic).  This guide assumes you already have your Java source files or jar files located on the lab computers and are familiar with Linux. If not, please see the [Transferring Files](http://sjengle.cs.usfca.edu/resources/transferring-files) and [Using Linux via Command Line](http://sjengle.cs.usfca.edu/resources/using-linux-via-command-line) guides before starting.  **Basics**  **Step 1: Compile**  Make sure you are currently in the directory with your source code files. Remember, Java source code is always stored a \*.java file. You first need to compile these files using Java. To do this, run the following command:  javac \*.java  Now, when you ls your directory, you should notice you also have \*.class files. Those are the files necessary to execute your Java code.  **Step 2: Execute**  To run your code, you need to know which class has the main() method. To execute your code, run the following command:  java -cp "." ClassName  Replace ClassName with the classname (no file extension) that contains your main() method. The -cp "." part tells the java tool to look for your \*.class files in the current directory.  **Arguments**  If you want to provide command-line arguments to your program, just include those arguments after the class name:  java -cp "." ClassName arg1 arg2 arg3 ...  Replace arg1 arg2 arg3 ... with your command-line arguments (separated by spaces). For example, suppose Driver takes a filename as an argument. To execute your code with the argument, run the following command:  java -cp "." Driver hello.txt  If you must specify a file name with a space inside it, use quotes around the filename:  java -cp "." Driver "Hello World.txt"  **Jar Files**  If you have already packaged your \*.class files in a \*.jar file, you can skip compiling your code with javac. To execute your code, run the following command:  java -cp filename.jar ClassName arguments  Replace filename.jar with the name of your jar file (assuming it is in the current directory). For this to work, your \*.jar file must contain ALL of the necessary classes for your program to run.  **Class Path**  You can set your $CLASSPATH variable on the lab computers so you no longer have to specify the -cp argument. This becomes especially important when you start using third-party libraries and must include multiple jar files. To do this, change to your home directory (run cd ~ at the prompt) and open the .bash\_profile file. If it does not exist yet, go ahead and create it. Add the following line to the file:  export CLASSPATH=".:.\*"  Some classes also provide some third-party libraries for you on the lab computers. For example, the CS 212 course places third-party libraries at /home/public/cs212/libraries/ on the lab computers. You can also add that directory (or any other) to your $CLASSPATH variable by adding the additional line:  export CLASSPATH="$CLASSPATH:/home/public/cs212/libraries/\*"  Save the file and exit. Then, run the following command:  . .bash\_profile  Make sure you include the . dot followed by a space before the .bash\_profile above. To test that everything is setup, run the following command:  echo $CLASSPATH  You should see all of the directories you specified in your $CLASSPATH variable. Now, to run Java programs, you can just do:  java ClassName arguments  Assuming that the necessary \*.class and \*.jar files are located in one of the directories you specified in your $CLASSPATH variable. |

**InetAddress** addr = **InetAddress**.getLocalHost();  
        
        *//Getting IPAddress of localhost - getHostAddress return IP Address*

*// in textual format*  
        **String** ipAddress = addr.getHostAddress();

Read more: <http://javarevisited.blogspot.com/2012/09/java-program-to-find-ip-address-of.html#ixzz2jnbBex9j>

InetAddress   in  = InetAddress.getLocalHost();

InetAddress[] all = InetAddress.getAllByName(in.getHostName());

for (int i=0; i<all.length; i++) {

System.out.println("  address = " + all[i]);

}

Normally, you can use servletRequest.getRemoteAddr() to get the client’s IP address that’s accessing your Java web application.

String ipAddress = request.getRemoteAddr();

But, if user is behind a proxy server or access your web server through a load balancer (for example, in cloud hosting), the above code will get the IP address of the proxy server or load balancer server, not the original IP address of a client.

To solve it, you should get the IP address of the request’s HTTP header “[X-Forwarded-For (XFF)](http://en.wikipedia.org/wiki/X-Forwarded-For)“.

*//is client behind something?*

String ipAddress = request.getHeader("X-FORWARDED-FOR");

**if** (ipAddress == **null**) {

ipAddress = request.getRemoteAddr();

}

byte[] ip = {(byte)192, (byte)168, 0, 0}; // for 192.168.0.x addresses

for (int i = 1; i <= 254; i++)

{

    ip[3] = i;

    InetAddress address = InetAddress.getByAddress(ip);

    ...

}

  ======

**Java program to find host name and IP address of local and remote machine**

**Problem Statement:** Many times we need to find the host name and IP address of local and the remote machines. Most importantly I want to do this with a Java program.  
  
**Solution:** Here is the main class, I have named it as HostNameFinder and to test these methods I've another class called as HostNameFinderTest.

/\*\*

\*

\* Classname : HostNameFinder.java

\* <b>HostNameFinder<b><br>

\* <p>

\* This class helps to get the host name and the IP addresses of the local machine and

\* remote machines. It provides static methods that can be used to get the very specific

\* information such as host name or the IP address.

\* </p>

\*

\* @author Manohar Negi

\* @version 1.0

\* @since 1.0

\* @see

\* Copyright notice

\*

\* Revision History:

\*

\* Date By Version Comments

\* ---------------------------------------------------------------------------------

\* Aug 31, 2011 Manohar Negi 1.0 Initial Draft

\* Sep 01, 2011 Manohar Negi 1.1 Modified to include more methods

\*

\*\*/

package com.mani.util;

import java.net.InetAddress;

import java.net.UnknownHostException;

/\*\*

\* <p>

\* <b>Functional Description:</b>

\* <p>

\* Some Description of the file

\*

\* Created: Month, Day ,YYYY

\*

\* @author

\* @version

\*/

public class HostNameFinder {

public static String getMyHostName() {

String hostname = null;

try {

InetAddress addr = InetAddress.getLocalHost();

hostname = addr.getHostName();

System.out.println("Host Name = " + hostname);

} catch (UnknownHostException e) {

e.printStackTrace();

}

return hostname;

}

public static String getMyIPAddress() {

String ipAddress = null;

try {

InetAddress addr = InetAddress.getLocalHost();

String ipAddr = addr.getHostAddress();

System.out.println("IP Address = " + ipAddr.toString());

} catch (UnknownHostException e) {

e.printStackTrace();

}

return ipAddress;

}

public static String getIPByAddress(String address) {

String ipAddress = null;

try {

InetAddress addr = InetAddress.getByName(address);

String ipAddr = addr.getHostAddress();

System.out.println("IP Address = " + ipAddr.toString());

} catch (UnknownHostException e) {

e.printStackTrace();

}

return ipAddress;

}

public static String getHostNameByAdress(String address) {

String hostname = null;

try {

InetAddress addr = InetAddress.getByName(address);

hostname = addr.getHostName();

System.out.println("Host Name = " + hostname);

} catch (UnknownHostException e) {

e.printStackTrace();

}

return hostname;

}

}

Here is the test class.

/\*\*

\*

\* Classname : HostNameFinderTest.java

\* <b>HostNameFinderTest<b><br>

\* <p>

\* This class is to test the methods of HostNameFinder.class

\* </p>

\*

\* @author Manohar Negi

\* @version 1.0

\* @since 1.0

\* @see

\* Copyright notice

\*

\* Revision History:

\*

\* Date By Version Comments

\* ---------------------------------------------------------------------------------

\* Aug 31, 2011 Manohar Negi 1.0 Initial Draft

\* Sep 01, 2011 Manohar Negi 1.1 Modified to include more methods

\*

\*\*/

package com.mani.util;

public class HotmNameFinderTest {

public static void main(String[] args) {

/\* get the host name for local machine \*/

HostNameFinder.getMyHostName();

/\* get the IP for local machine \*/

HostNameFinder.getMyIPAddress();

/\* get the host name for localhost \*/

HostNameFinder.getHostNameByAdress("localhost");

/\* get the IP for localhost \*/

HostNameFinder.getIPByAddress("localhost");

/\* get the host name for www.google.com \*/

HostNameFinder.getHostNameByAdress("www.google.com");

/\* get the IP for www.google.com \*/

HostNameFinder.getIPByAddress("www.google.com");

}

}

# 15 Practical Linux cURL Command Examples (cURL Download Examples)

cURL is a software package which consists of command line tool and a library for transferring data using URL syntax.

cURL supports various protocols like, DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet and TFTP.

This article provides 15 practical cURL usage examples.

### 1. Download a Single File

The following command will get the content of the URL and display it in the STDOUT (i.e on your terminal).

$ curl http://www.centos.org

To store the output in a file, you an redirect it as shown below. This will also display some additional download statistics.

$ curl http://www.centos.org > centos-org.html

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 27329 0 27329 0 0 104k 0 --:--:-- --:--:-- --:--:-- 167k

### 2. Save the cURL Output to a file

We can save the result of the curl command to a file by using -o/-O options.

* -o (lowercase o) the result will be saved in the filename provided in the command line
* -O (uppercase O) the filename in the URL will be taken and it will be used as the filename to store the result

$ curl -o mygettext.html http://www.gnu.org/software/gettext/manual/gettext.html

Now the page gettext.html will be saved in the file named ‘mygettext.html’. You can also note that when running curl with -o option, it displays the progress meter for the download as follows.

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

66 1215k 66 805k 0 0 33060 0 0:00:37 0:00:24 0:00:13 45900

100 1215k 100 1215k 0 0 39474 0 0:00:31 0:00:31 --:--:-- 68987

When you use curl -O (uppercase O), it will save the content in the file named ‘gettext.html’ itself in the local machine.

$ curl -O http://www.gnu.org/software/gettext/manual/gettext.html

Note: When curl has to write the data to the terminal, it disables the Progress Meter, to avoid confusion in printing. We can use ‘>’|'-o’|'-O’ options to move the result to a file.

Similar to cURL, you can also use wget to download files. Refer to [wget examples](http://www.thegeekstuff.com/2009/09/the-ultimate-wget-download-guide-with-15-awesome-examples/) to understand how to use wget effectively.

### 3. Fetch Multiple Files at a time

We can download multiple files in a single shot by specifying the URLs on the command line.  
Syntax:

$ curl -O URL1 -O URL2

The below command will download both index.html and gettext.html and save it in the same name under the current directory.

$ curl -O http://www.gnu.org/software/gettext/manual/html\_node/index.html -O http://www.gnu.org/software/gettext/manual/gettext.html

Please note that when we download multiple files from a same sever as shown above, curl will try to re-use the connection.

### 4. Follow HTTP Location Headers with -L option

By default CURL doesn’t follow the HTTP Location headers. It is also termed as Redirects. When a requested web page is moved to another place, then an HTTP Location header will be sent as a Response and it will have where the actual web page is located.

For example, when someone types google.com in the browser from India, it will be automatically redirected to ‘google.co.in’. This is done based on the HTTP Location header as shown below.

$ curl http://www.google.com

<TITLE>302 Moved</TITLE>

<H1>302 Moved</H1>

The document has moved

<A HREF="http://www.google.co.in/">here</A>

The above output says that the requested document is moved to ‘http://www.google.co.in/’.

We can insists curl to follow the redirection using -L option, as shown below. Now it will download the google.co.in’s html source code.

$ curl -L http://www.google.com

### 5. Continue/Resume a Previous Download

Using curl -C option, you can continue a download which was stopped already for some reason. This will be helpful when you download large files, and the download got interrupted.

If we say ‘-C -’, then curl will find from where to start resuming the download. We can also give an offset ‘-C <offset>’. The given offset bytes will be skipped from the beginning for the source file.

Start a big download using curl, and press Ctrl-C to stop it in between the download.

$ curl -O http://www.gnu.org/software/gettext/manual/gettext.html

############## 20.1%

Note: -# is used to display a progress bar instead of a progress meter.

Now the above download was stopped at 20.1%. Using “curl -C -”, we can continue the download from where it left off earlier. Now the download continues from 20.1%.

curl -C - -O http://www.gnu.org/software/gettext/manual/gettext.html

############### 21.1%

### 6. Limit the Rate of Data Transfer

You can limit the amount at which the data gets transferred using –limit-rate option. You can specify the maximum transfer rate as argument.

$ curl --limit-rate 1000B -O http://www.gnu.org/software/gettext/manual/gettext.html

The above command is limiting the data transfer to 1000 Bytes/second. curl may use higher transfer rate for short span of time. But on an average, it will come around to 1000B/second.

The following was the progress meter for the above command. You can see that the current speed is near to the 1000 Bytes.

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

1 1215k 1 13601 0 0 957 0 0:21:40 0:00:14 0:21:26 999

1 1215k 1 14601 0 0 960 0 0:21:36 0:00:15 0:21:21 999

1 1215k 1 15601 0 0 962 0 0:21:34 0:00:16 0:21:18 999

### 7. Download a file only if it is modified before/after the given time

We can get the files that are modified after a particular time using -z option in curl. This will work for both FTP & HTTP.

$ curl -z 21-Dec-11 http://www.example.com/yy.html

The above command will download the yy.html only if it is modified later than the given date and time

$ curl -z -21-Dec-11 http://www.example.com/yy.html

The above command will download the yy.html, if it is modified before than the given date and time.

Please refer ‘man curl\_getdate’ for the various syntax supported for the date expression

### 8. Pass HTTP Authentication in cURL

Sometime, websites will require a username and password to view the content ( can be done with .htaccess file ). With the help of -u option, we can pass those credentials from cURL to the web server as shown below.

$ curl -u username:password URL

Note: By default curl uses Basic HTTP Authentication. We can specify other authentication method using –ntlm | –digest.

### 9. Download Files from FTP server

cURL can also be used to download files from FTP servers. If the given FTP path is a directory, by default it will list the files under the specific directory.

$ curl -u ftpuser:ftppass -O ftp://ftp\_server/public\_html/xss.php

The above command will download the xss.php file from the ftp server and save it in the local directory.

$ curl -u ftpuser:ftppass -O ftp://ftp\_server/public\_html/

Here, the given URL refers to a directory. So cURL will list all the files and directories under the given URL

If you are new to FTP/sFTP, refer [ftp sftp tutorial for beginners.](http://www.thegeekstuff.com/2010/06/ftp-sftp-tutorial/)

### 10. List/Download using Ranges

cURL supports ranges to be given in the URL. When a range is given, files matching within the range will be downloaded. It will be helpful to download packages from the FTP mirror sites.

$ curl ftp://ftp.uk.debian.org/debian/pool/main/[a-z]/

The above command will list out all the packages from a-z ranges in the terminal.

### 11. Upload Files to FTP Server

Curl can also be used to upload files to the FTP server with -T option.

$ curl -u ftpuser:ftppass -T myfile.txt ftp://ftp.testserver.com

The above command will upload the file named myfile.txt to the FTP server. You can also upload multiple files at a same time using the range operations.

$ curl -u ftpuser:ftppass -T "{file1,file2}" ftp://ftp.testserver.com

Optionally we can use “.” to get the input from STDIN and transfer to the remote.

$ curl -u ftpuser:ftppass -T - ftp://ftp.testserver.com/myfile\_1.txt

The above command will get the input from the user from Standard Input and save the contents in the ftp server under the name ‘myfile\_1.txt’.

You can provide one ‘-T’ for each URL and the pair specifies what to upload where.

### 12. More Information using Verbose and Trace Option

You can get to know what is happening using the -v option. -v option enable the verbose mode and it will print the details

curl -v http://google.co.in

The about command will output the following

\* About to connect() to www.google.co.in port 80 (#0)

\* Trying 74.125.236.56... connected

\* Connected to www.google.co.in (74.125.236.56) port 80 (#0)

> GET / HTTP/1.1

> User-Agent: curl/7.21.0 (i486-pc-linux-gnu) libcurl/7.21.0 OpenSSL/0.9.8o zlib/1.2.3.4 libidn/1.15 libssh2/1.2.6

> Host: www.google.co.in

> Accept: \*/\*

>

\* HTTP 1.0, assume close after body

< HTTP/1.0 200 OK

< Date: Tue, 10 Apr 2012 11:18:39 GMT

< Expires: -1

< Cache-Control: private, max-age=0

< Content-Type: text/html; charset=ISO-8859-1

< Set-Cookie: PREF=ID=7c497a6b15cc092d:FF=0:TM=1334056719:LM=1334056719:S=UORpBwxFmTRkbXLj; expires=Thu, 10-Apr-2014 11:18:39 GMT; path=/; domain=.google.co.in

.

.

If you need more detailed information then you can use the –trace option. The trace option will enable a full trace dump of all incoming/outgoing data to the given file

=> Send header, 169 bytes (0xa9)

0000: 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a GET / HTTP/1.1..

0010: 55 73 65 72 2d 41 67 65 6e 74 3a 20 63 75 72 6c User-Agent: curl

..

0060: 2e 32 2e 33 2e 34 20 6c 69 62 69 64 begin\_of\_the\_skype\_highlighting 69 62 69 64 FREE  end\_of\_the\_skype\_highlighting 6e 2f 31 2e .2.3.4 libidn/1.

0070: 31 35 20 6c 69 62 73 73 68 32 2f 31 2e 32 2e 36 15 libssh2/1.2.6

0080: 0d 0a 48 6f 73 74 3a 20 77 77 77 2e 67 6f 6f 67 ..Host: www.goog

0090: 6c 65 2e 63 6f 2e 69 6e 0d 0a 41 63 63 65 70 74 le.co.in..Accept

00a0: 3a 20 2a 2f 2a 0d 0a 0d 0a : \*/\*....

== Info: HTTP 1.0, assume close after body

<= Recv header, 17 bytes (0x11)

0000: 48 54 54 50 2f 31 2e 30 20 32 30 30 20 4f 4b 0d HTTP/1.0 200 OK.

0010: 0a

This verbose and trace option will come in handy when curl fails due to some reason and we don’t know why.

### 13. Get Definition of a Word using DICT Protocol

You can use cURL to get the definition for a word with the help of DICT protocol. We need to pass a Dictionary Server URL to it.

$ curl dict://dict.org/d:bash

The above command will list the meaning for bash as follows

151 "Bash" gcide "The Collaborative International Dictionary of English v.0.48"

Bash \Bash\, v. t. [imp. & p. p. {Bashed}; p. pr. & vb. n.

{Bashing}.] [Perh. of imitative origin; or cf. Dan. baske to

strike, bask a blow, Sw. basa to beat, bas a beating.]

To strike heavily; to beat; to crush. [Prov. Eng. & Scot.]

--Hall Caine.

[1913 Webster]

Bash her open with a rock. --Kipling.

[Webster 1913 Suppl.]

.

151 "Bash" gcide "The Collaborative International Dictionary of English v.0.48"

Bash \Bash\, n.

1. a forceful blow, especially one that does damage to its

target.

[PJC]

.

Now you can see that it uses “The Collaborative International Dictionary of English”. There are many dictionaries are available. We can list all the dictionaries using

$ curl dict://dict.org/show:db

jargon "The Jargon File (version 4.4.7, 29 Dec 2003)"

foldoc "The Free On-line Dictionary of Computing (26 July 2010)"

easton "Easton's 1897 Bible Dictionary"

hitchcock "Hitchcock's Bible Names Dictionary (late 1800's)"

bouvier "Bouvier's Law Dictionary, Revised 6th Ed (1856)"

Now in-order to find the actual meaning of Bash in computer we can search for bash in “foldoc” dictionary as follows

$ curl dict://dict.org/d:bash:foldoc

The result will be,

bash

Bourne Again SHell. {GNU}'s {command interpreter} for {Unix}.

Bash is a {Posix}-compatible {shell} with full {Bourne shell}

syntax, and some {C shell} commands built in. The Bourne

Again Shell supports {Emacs}-style command-line editing, job

control, functions, and on-line help. Written by Brian Fox of

{UCSB}.

For more details with regard to DICT please read [RFC2229](http://tools.ietf.org/html/rfc2229)

### 14. Use Proxy to Download a File

We can specify cURL to use proxy to do the specific operation using -x option. We need to specify the host and port of the proxy.

$ curl -x proxysever.test.com:3128 http://google.co.in

### 15. Send Mail using SMTP Protocol

cURL can also be used to send mail using the SMTP protocol. You should specify the from-address, to-address, and the mailserver ip-address as shown below.

$ curl --mail-from blah@test.com --mail-rcpt foo@test.com smtp://mailserver.com

Once the above command is entered, it will wait for the user to provide the data to mail. Once you’ve composed your message, type . (period) as the last line, which will send the email immediately.

### What is the Ping Command?:

The ping [command](http://pcsupport.about.com/od/termsc/g/commands.htm) is a [Command Prompt command](http://pcsupport.about.com/od/commandlinereference/tp/command-prompt-commands-p1.htm) used to test the ability of the source computer to reach a specified destination computer. The ping command is usually used as a simple way verify that a computer can communicate over the network with another computer or network device.

The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response. How many of those responses are returned, and how long it takes for them to return, are the two major pieces of information that the ping command provides.

### Ping Command Syntax:

**ping** [**-t**] [**-a**] [**-n** *count*] [**-l** *size*] [**-f**] [**-i** *TTL*] [**-v** *TOS*] [**-r** *count*] [**-s** *count*] [**-w** *timeout*] [**-R**] [**-S** *srcaddr*] [**-4**] [**-6**] *target* [**/?**]

**Tip:** See [How To Read Command Syntax](http://pcsupport.about.com/od/commandlinereference/a/command-syntax.htm) if you're not sure how to interpret the ping command [syntax](http://pcsupport.about.com/od/termss/g/syntax.htm) above.

**-t** = Using this option will ping the *target* until you force it to stop using [Ctrl-C](http://pcsupport.about.com/od/termsc/g/ctrl-c.htm).

**-a** = This ping command option will resolve, if possible, the [hostname](http://pcsupport.about.com/od/termshm/g/hostname.htm) of an [IP address](http://pcsupport.about.com/od/termsi/g/ip-address.htm) *target*.

**-n** *count* = This option sets the number of ICMP Echo Request messages to send. If you execute the ping command without this option, four requests will be sent.

**-l** *size* = Use this option to set the size, in bytes, of the echo request packet from 32 to 65,527. The ping command will send a 32 byte echo request if you don't use the **-l** option.

**-f** = Use this ping command option to prevent ICMP Echo Requests from being fragmented by routers between you and the *target*. The **-f** option is most often used to troubleshoot Path Maximum Transmission Unit (PMTU) issues.

**-i** *TTL* = This option sets the Time to Live (TTL) value, the maximum of which is 255.

**-v** *TOS* = This option allows you to set a Type of Service (TOS) value. Beginning in Windows 7, this option no longer functions but still exists for compatibility reasons.

**-r** *count* = Use this ping command option to specify the number of [hops](http://pcsupport.about.com/od/termshm/g/hop.htm) between the your computer and the *target* computer or device that you'd like to be recorded and displayed. The maximum value for *count* is 9 so use the [tracert command](http://pcsupport.about.com/od/commandlinereference/p/tracert-command.htm) instead if you're interested in viewing all hops between two devices.

**-s** *count* = Use this option to report the time, in Internet Timestamp format, that each echo request is received and echo reply is sent. The maximum value for *count* is 4 meaning that only the first four hops can be time stamped.

**-w** *timeout* = Specifying a *timeout* value when executing the ping command adjusts the amount of time, in milliseconds, that ping waits for each reply. If you don't use the **-w** option, the default timeout value is used which is 4000, or 4 seconds.

**-R** = This option tells the ping command to trace the round trip path.

**-S** *srcaddr* = Use this option to specify the source address.

**-4** = This forces the ping command to use IPv4 only but is only necessary if *target* is a hostname and not an IP address.

**-6** = This forces the ping command to use IPv6 only but as with the **-4** option, is only necessary when pinging a hostname.

*target* = This is the destination you wish to ping, either an IP address or a hostname.

**/?** = Use the [help switch](http://pcsupport.about.com/od/termshm/g/help-switch.htm) with the ping command to show detailed help about the command's several options.

**Note:** The **-f**, **-v**, **-r**, **-s**, **-j**, and **-k** options work when pinging IPv4 addresses only. The **-R** and **-S** options only work with IPv6.

Other less commonly used switches for the ping command exist including [**-j** *host-list*] and [**-k** *host-list*]. Execute **ping /?** from the Command Prompt for more information on these two options.

**Tip:** Save all that ping command output to a file using a [redirection operator](http://pcsupport.about.com/od/termsr/a/redirection-operator.htm). See [How To Redirect Command Output to a File](http://pcsupport.about.com/od/commandlinereference/a/redirect-command-output-to-file.htm) for instructions or see my [Command Prompt Tricks](http://pcsupport.about.com/od/commandlinereference/tp/command-prompt-tricks-hacks.htm) list for more tips.

### Ping Command Examples:

**ping -n 5 -l 1500 www.google.com**

In this example, the ping command is used to ping the hostname *www.google.com*. The **-n** option tells the ping command to send *5* ICMP Echo Requests instead of the default of 4 and the **-l** option sets the packet size for each request to *1500* bytes instead of the default of 32 bytes. The result displayed in the Command Prompt window will look something like this:

Pinging www.google.com [74.125.224.82] with 1500 bytes of data:

Reply from 74.125.224.82: bytes=1500 time=68ms TTL=52

Reply from 74.125.224.82: bytes=1500 time=68ms TTL=52

Reply from 74.125.224.82: bytes=1500 time=65ms TTL=52

Reply from 74.125.224.82: bytes=1500 time=66ms TTL=52

Reply from 74.125.224.82: bytes=1500 time=70ms TTL=52

Ping statistics for 74.125.224.82:

Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 65ms, Maximum = 70ms, Average = 67ms

The *0% loss* reported under *Ping statistics for 74.125.224.82* tells me that each ICMP Echo Request message sent to *www.google.com* was returned. This means that, as far as my network connection goes, I can communicate with Google's website just fine.

**ping 127.0.0.1**

In the above example, I'm pinging *127.0.0.1*, also called the IPv4 localhost IP address or IPv4 loopback IP address, without options.

Using the ping command to ping *127.0.0.1* is an excellent way to test that Windows' network features are working properly but it says nothing about your own network hardware or your connection to any other computer or device. The IPv6 version of this test would be **ping ::1**.

**ping -a 192.168.1.22**

In this example I'm asking the ping command to find the hostname assigned to the *192.168.1.22* IP address but otherwise ping it as normal.

Pinging J3RTY22 [192.168.1.22] with 32 bytes of data:

Reply from 192.168.1.22: bytes=32 time<1ms TTL=64

Reply from 192.168.1.22: bytes=32 time<1ms TTL=64

Reply from 192.168.1.22: bytes=32 time=1ms TTL=64

Reply from 192.168.1.22: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.22:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 1ms, Average = 0ms

As you can see, the ping command resolved the IP address I entered, *192.168.1.22*, as the hostname *J3RTY22* and then executed the remainder of the ping with default settings.

**ping -t -6 SERVER**

In this example, I force the ping command to use IPv6 with the **-6** option and continue to ping *SERVER* indefinitely with the **-t** option.

Pinging SERVER [fe80::fd1a:3327:2937:7df3%10] with 32 bytes of data:

Reply from fe80::fd1a:3327:2937:7df3%10: time=1ms

Reply from fe80::fd1a:3327:2937:7df3%10: time<1ms

Reply from fe80::fd1a:3327:2937:7df3%10: time<1ms

Reply from fe80::fd1a:3327:2937:7df3%10: time<1ms

Reply from fe80::fd1a:3327:2937:7df3%10: time<1ms

Reply from fe80::fd1a:3327:2937:7df3%10: time<1ms

Reply from fe80::fd1a:3327:2937:7df3%10: time<1ms

Ping statistics for fe80::fd1a:3327:2937:7df3%10:

Packets: Sent = 7, Received = 7, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 1ms, Average = 0ms

Control-C

^C

I interrupted the ping manually with Ctrl-C after seven replies. Also, as you can see, the **-6** option produced IPv6 addresses.

**Tip:** The number after the % in the replies generated in this ping command example is the IPv6 Zone ID, which most often indicates the network interface used. You can generate a table of Zone IDs matched with your network interface names by executing **netsh interface ipv6 show interface**. The IPv6 Zone ID is the number in the *Idx* column.