curl

Transfer data from or to a server, using one of the protocols: HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP or FILE.
(To transfer multiple files use wget or FTP.)

```
Syntax
      curl [options] [URL...]

Key
   url  One or multiple URLs that will be fetched in sequence.

        Multiple URLs or parts of URLs can be specified by writing part sets within braces as in:

        https://site.{one,two,three}.com

        or get sequences of alphanumeric series by using [] as in:

        ftp://ftp.numericals.com/file[1-100].txt
        ftp://ftp.numericals.com/file[001-100].txt (with leading zeros)
        ftp://ftp.letters.com/file[a-z].txt

   -A "agent string"
   --user-agent "agent string"
        Specify the User-Agent string to send to the HTTP server.
        To encode blanks in the string, surround the string with single
        quote marks. This can also be set with -H, --header option. (HTTP)

   -b name=data
   --cookie name=data
        Send the data to the HTTP server as a cookie. It is supposedly the data
        previously received from the server in a "Set-Cookie:" line.
        The data should be in the format  "NAME1=VALUE1; NAME2=VALUE2".

   -c filename
   --cookie-jar file name
        Save cookies to file after a completed operation.
        Curl writes all cookies previously read from a specified file
        as well as all cookies received from remote server(s).
        If no cookies are known, no file will be written.
        TO write to stdout, set the file name to a single dash,  "-"

   --compressed
          Request a compressed response using one of the algorithms
        curl supports (gzip), and save  the  uncompressed  document.
        If this option  is  used  and  the server sends an unsupported encoding,
        curl will report an error.(HTTP)

   -d @file
   -d "string"
   --data "string"
        Send the specified data in an (HTTP) POST request, in the same way that a
        web browser does.
        This  will pass the data using the content-type
        application/x-www-form-urlencoded.  Compare to -F, --form.

        -d, --data is the same as  --data-ascii.  To post data in pure binary, use --data-binary.
        To URL-encode the value of a form field you may use --data-urlencode.

        Multiple date options will be merged together.
        Thus,  using  '-d  name=daniel  -d skill=lousy' would generate a post
        that looks like  'name=daniel&skill=lousy'.

        If the data starts with @, the rest should be a filename containing the data.

   -f, --fail
        (HTTP) Fail silently (no output at all) on server errors.
        This is mostly done to enable scripts etc to better deal with failed attempts.
        In normal cases when an HTTP server fails to deliver a document, it returns an HTML document
        stating so (which often also describes why and more).
        This flag will prevent curl from outputting that and return error 22.

        This method is not fail-safe and there are occasions where non-successful response codes will
        slip through, especially when authentication is involved (response codes 401 and 407).
        See also --fail-with-body.
        Example:
        curl --fail https://example.com
```

```
-F name=@file
-F name=content
  --form name=content
     Emulate a filled-in form in which a user has pressed the submit button.
     This will POST data using the Content-Type  multipart/form-data according
     to  RFC 2388.  This enables uploading of binary files etc.

     If the data starts with @, the rest should be a filename.
     To just get the content part from a file, prefix the file name with the
     symbol <. The difference between @ and <  is that  @  makes a file get
     attached in the post as a file upload, while the < makes a text field
     and gets the contents for that text field from a file.

     Example, to send your password file to the server, where 'password' is
     the name of the form-field to which /etc/passwd will be the input:

     curl -F password=@/etc/passwd www.mypasswords.com

-H "name: value"
--header "name: value"
     Extra header to include in the request when sending HTTP to a server.
     You may specify any number of extra headers.
     curl -H "User-Agent: yes-please/2000" https://example.com

-I
--head
     Fetch the HTTP-header only! (HTTP/FTP/FILE)
     HTTP-servers feature the  command  HEAD which this uses to get nothing but
     the header of a document. When used on an FTP or FILE file, curl displays
     the file size and last modification time only.

-k
--insecure
     This option explicitly allows curl to  perform  "insecure" SSL connections
     and transfers. All SSL connections are attempted in secure mode using the
     CA certificate  bundle  installed by  default. This  makes  all connections
     considered "insecure" fail unless -k, --insecure is used.(SSL)

--limit-rate speed
     Specify the maximum transfer rate.  This feature is useful if you have a
     limited pipe and you'd like your transfer not to use your entire bandwidth.

     The given speed is measured in bytes/second, unless a suffix  is appended.
     Appending 'k' or 'K' will count the number as kilobytes/sec, 'm' or M' megabytes,
     while 'g' or 'G' makes it gigabytes/sec. Eg: 200K, 3m, 1G.


-L, --location (HTTP)
     If the server reports that the requested page has moved to a different location
     (indicated with a Location: header and a 3XX response code), this option will
     make curl redo the request on the new place.

-m seconds
--max-time seconds
     Maximum time that you allow the whole operation to take.
     This is useful for preventing your batch jobs from  hanging for hours due to
     slow networks or links going down.  See also the --connect-timeout option.

-o file
--output file
     Write output to file instead of stdout. If you are using {} or [] to fetch
     multiple documents, you can use '#' followed by a number  in  the file specifier.
     That variable will be replaced with the current string for the URL being fetched.
     Like in:
              curl https://{one,two}.site.com -o "file_#1.txt"
     or use several variables like:
              curl https://{site,host}.host[1-5].com -o "#1_#2"
     You may use this option as many times as the number of URLs you have.

     See also --create-dirs option to create the local directories dynamically.
     Specify '-' to force the output to stdout.

-O
--remote-name
     Write output to a local file named like the remote file we get.
     (Only the file part of the remote file is used, the path is cut off.)
     The  remote file name to use for saving is extracted from the given URL, nothing else.
     Consequentially, the file will be saved in the  current  working directory.

-s
--silent
     Silent or quiet mode. Don't show progress meter or error messages.
```

```
-S, --show-error
     When used with -s, --silent, it makes curl show an error message if it fails.
     This option is global and does not need to be specified for each use of -:, --next.
     Example:
     curl --show-error --silent https://example.com

--trace-ascii file
     Enable a full trace dump of all  incoming  and  outgoing  data, including
     descriptive information, to the given output file.
     Use "-" as filename to have the output sent to stdout.
     This option overrides previous uses of -v, --verbose or --trace-ascii.
     If this option is used several times, the last one will be used.

-T file
  --upload-file file
     Transfer the specified local file to the remote  URL.  PUT
     If there is no file part in the specified URL, Curl will append the local
     file name. You must use a trailing / on the last directory to really prove
     to Curl that there is no file name or curl will think that the last
     directory name is the remote file name to use.
     Use the file name "-" to use stdin

     You can specify one -T for each URL on the command line. Each -T
     + URL pair specifies what to upload and to where. curl also supports "globbing"
     of the -T argument, meaning that you can upload multiple files to a single URL
     like this:

     curl -T "{file1,file2}" https://www.uploadtothissite.com
     or even
     curl -T "img[1-1000].png" ftp://ftp.picturemania.com/upload/

-u user:password
--user user:password
     The username and password to use for server authentication.
     Overrides -n, --netrc and --netrc-optional.

     If you just give the user name (without entering a  colon) curl will prompt
     for a password.

     If you use an SSPI-enabled curl binary and do NTLM authentication, you can
     force curl to pick up the username and password from your environment by
     specifying a single colon with this option: "-u :".

     If this option is used several times, the last one will be used.

-v
--verbose
     Make more verbose/talkative. Mostly useful for debugging.

-w
--write-out format
     Define extra info to display on stdout after a completed and successful operation.
     The format is a string that may contain plain text mixed with any
     number of variables. The format string can be specified  as "string", or to
     read from a file specify  "@filename"
     to read the format from stdin use "@-".

     Various variables may be included in the format and will be substituted by
     curl (file size, ip address etc see man curl for details).
     variables are specified as %{variable_name}

     Output a newline using \n, a carriage return with \r and a tab space with \t.

-x host:port
-x  [protocol://][user:password@]proxyhost[:port]
--proxy [protocol://][user:password@]proxyhost[:port]
     Use the  specified HTTP proxy.
     If the port number is not specified, it is assumed at port 1080.
```

This page is a heavily abbreviated selection of the full options, for more detail including return codes, run `man  curl`

curl is a powerful tool, please use it responsibly, regularly and repeatedly downloading files from the same website can result in your IP being automatically flagged/blocked.

The return status is zero if no errors occur, non-zero otherwise.

Examples

Retrieve a web page and display in the terminal, use --include (-i) option to also display header information:

```
$ curl https://ss64.com
$ curl https://ss64.com -i
```

Retrieve a web page and save to a file

```
$ curl https://ss64.com/bash/ -o ss64.html
```

Retrieve a web page, or its redirected target:

```
$ curl ss64.co/bash/
$ curl ss64.co/bash/ --location
```

Limit the rate of data transfer to 1 Kilobytes/sec:

```
$ curl https://ss64.com/bash/ --limit-rate 1k -o ss64.html
```

Retrieve a web page, passing a specific User-Agent HTTP header (some websites use this to sniff the browser used):

```
$ curl -A "Mozilla FireFox(72.0)" https://example.com
```

Download via a proxy server:

```
$ curl -x proxy.example.com:3128 https://ss64.com/bash/
```

"We are shallow because we have become enslaved by gross materialism, the glitter of gold and its equivalents, for which reason we think that only the material goods of this earth can satisfy us and we must therefore grab as much as can while we are able" ~ F. Sionil Jose

Related linux commands

HTTPie - http command for testing and debugging HTTP servers.
ftp - File Transfer Protocol.
Curl project downloads - Official source which may be more up to date than other packaged copies.