WGET

NAME

Wget - The non-interactive network downloader.

SYNOPSIS

wget [*option*]... [*URL*]...

DESCRIPTION

GNU Wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTPproxies.

Wget is non-interactive, meaning that it can work in the background, while the user is not logged on. This allows you to start a retrieval and disconnect from the system, letting Wget finish the work. By contrast, most of the Web browsers require constant user's presence, which can be a great hindrance when transferring a lot of data.

Wget can follow links in HTML, XHTML, and CSS pages, to create local versions of remote web sites, fully recreating the directory structure of the original site. This is sometimes referred to as "recursive downloading." While doing that, Wget respects the Robot Exclusion Standard (*/robots.txt*). Wget can be instructed to convert the links in downloaded files to point at the local files, for offline viewing.

Wget has been designed for robustness over slow or unstable network connections; if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. If the server supports regetting, it will instruct the server to continue the download from where it left off.

OPTIONS

Option Syntax
Since Wget uses GNU getopt to process command-line arguments, every option has a long form along with the short one. Long options are more convenient to remember, but take time to type. You may freely mix different option styles, or specify options after the command-line arguments. Thus you may write:

```
wget -r --tries=10 http://fly.srk.fer.hr/ -o log
```

The space between the option accepting an argument and the argument may be omitted. Instead of **-o log** you can write **-olog**.

You may put several options that do not require arguments together, like:

```
wget -drc <URL>
```

This is completely equivalent to:

```
wget -d -r -c <URL>
```

Since the options can be specified after the arguments, you may terminate them with **--**. So the following will try to download URL **-x**, reporting failure to *log*:

```
wget -o log -- -x
```

The options that accept comma-separated lists all respect the convention that specifying an empty list clears its value. This can be useful to clear the *.wgetrc*settings. For instance, if your *.wgetrc* sets `"exclude_directories"` to */cgi-bin*, the following example will first reset it, and then set it to exclude */~nobody* and*/~somebody*. You can also clear the lists in *.wgetrc*.

```
wget -X " -X /~nobody,/~somebody
```

Most options that do not accept arguments are *boolean* options, so named because their state can be captured with a yes-or-no ("boolean") variable. For example, **--follow-ftp** tells Wget to follow FTP links from HTML files and, on the other hand, **--no-glob** tells it not to perform file globbing on FTP URLs. A boolean option is either *affirmative* or *negative* (beginning with **--no**). All such options share several properties.

Unless stated otherwise, it is assumed that the default behavior is the opposite of what the option accomplishes. For example, the documented existence of **--follow-ftp** assumes that the default is to *not* follow FTP links from HTML pages.

Affirmative options can be negated by prepending the **--no-** to the option name; negative options can be negated by omitting the **--no-** prefix. This might seem superfluous---if the default for an affirmative option is to not do something, then why provide a way to explicitly turn it off? But the startup file may in fact change the default. For instance, using `"follow_ftp = on"` in *.wgetrc* makes

Wget *follow* FTP links by default, and using **--no-follow-ftp** is the only way to restore the factory default from the command line.

Basic Startup Options

-V
--version
Display the version of Wget.

-h
--help
Print a help message describing all of Wget's command-line options.

-b
--background
Go to background immediately after startup. If no output file is specified via the **-o**, output is redirected to *wget-log*.

-e *command*
--execute *command*
Execute *command* as if it were a part of *.wgetrc*. A command thus invoked will be executed *after* the commands in *.wgetrc*, thus taking precedence over them. If you need to specify more than one wgetrc command, use multiple instances of **-e**.

Logging and Input File Options

-o *logfile*
--output-file=*logfile*
Log all messages to *logfile*. The messages are normally reported to standard error.

-a *logfile*
--append-output=*logfile*
Append to *logfile*. This is the same as **-o**, only it appends to *logfile* instead of overwriting the old log file. If *logfile* does not exist, a new file is created.

-d
--debug
Turn on debug output, meaning various information important to the developers of Wget if it does not work properly. Your system administrator may have chosen to compile Wget without debug support, in which case **-d** will not work. Please note that compiling with debug support is always safe---Wget compiled with the debug support will *not*print any debug info unless requested with **-d**.

-q
--quiet
Turn off Wget's output.

-v
--verbose
Turn on verbose output, with all the available data. The default output is verbose.

-nv
--no-verbose
Turn off verbose without being completely quiet (use **-q** for that), which means that error messages and basic information still get printed.

--report-speed=*type*

Output bandwidth as *type*. The only accepted value is **bits**.

-i *file*

--input-file=*file*

Read URLs from a local or external *file*. If **-** is specified as *file*, URLs are read from the standard input. (Use **./-** to read from a file literally named**-**.)

If this function is used, no URLs need be present on the command line. If there are URLs both on the command line and in an input file, those on the command lines will be the first ones to be retrieved. If **--force-html** is not specified, then *file* should consist of a series of URLs, one per line.

However, if you specify **--force-html**, the document will be regarded as **html**. In that case you may have problems with relative links, which you can solve either by adding `"<base href="`*url*`">"` to the documents or by specifying **--base**=*url* on the command line.

If the *file* is an external one, the document will be automatically treated as **html** if the Content-Type matches **text/html**. Furthermore, the *file*'s location will be implicitly used as base href if none was specified.

--input-metalink=*file*

Downloads files covered in local Metalink *file*. Metalink version 3 and 4 are supported.

--keep-badhash

Keeps downloaded Metalink's files with a bad hash. It appends .badhash to the name of Metalink's files which have a checksum mismatch, except without overwriting existing files.

--metalink-over-http

Issues HTTP HEAD request instead of GET and extracts Metalink metadata from response headers. Then it switches to Metalink download. If no valid Metalink metadata is found, it falls back to ordinary HTTPdownload. Enables **Content-Type: application/metalink4+xml** files download/processing.

--metalink-index=*number*

Set the Metalink **application/metalink4+xml** metaurl ordinal NUMBER.From 1 to the total number of "application/metalink4+xml" available. Specify 0 or **inf** to choose the first good one. Metaurls, such as those from a **--metalink-over-http**, may have been sorted by priority key's value; keep this in mind to choose the right NUMBER.

--preferred-location

Set preferred location for Metalink resources. This has effect if multiple resources with same priority are available.

--xattr

Enable use of file system's extended attributes to save the original URLand the Referer HTTP header value if used.

Be aware that the URL might contain private information like access tokens or credentials.

-F

--force-html

When input is read from a file, force it to be treated as an HTML file. This enables you to retrieve relative links from existing HTML files on your local disk, by adding `"<base href="`*url*`">"` to HTML, or using the **--base**command-line option.

-B *URL*

--base= *URL*

Resolves relative links using *URL* as the point of reference, when reading links from an HTML file specified via the **-i/--input-file** option (together with **--force-html**, or when the input file was fetched remotely from a server describing it as HTML). This is equivalent to the presence of a `"BASE"` tag in the HTML input file, with *URL* as the value for the `"href"`attribute.

For instance, if you specify **http://foo/bar/a.html** for *URL* , and Wget reads **../baz/b.html** from the input file, it would be resolved to **http://foo/baz/b.html**.

--config= *FILE*

Specify the location of a startup file you wish to use instead of the default one(s). Use --no-config to disable reading of config files. If both --config and --no-config are given, --no-config is ignored.

--rejected-log=*logfile*

Logs all URL rejections to *logfile* as comma separated values. The values include the reason of rejection, the URL and the parent URL it was found in.

Download Options

--bind-address= *ADDRESS*

When making client TCP/IP connections, bind to *ADDRESS* on the local machine. *ADDRESS* may be specified as a hostname or IP address. This option can be useful if your machine is bound to multiple IPs.

--bind-dns-address= *ADDRESS*

[libcares only] This address overrides the route for DNS requests. If you ever need to circumvent the standard settings from /etc/resolv.conf, this option together with **--dns-servers** is your friend. *ADDRESS* must be specified either as IPv4 or IPv6 address. Wget needs to be built with libcares for this option to be available.

--dns-servers= *ADDRESSES*

[libcares only] The given address(es) override the standard nameserver addresses, e.g. as configured in /etc/resolv.conf. *ADDRESSES* may be specified either as IPv4 or IPv6 addresses, comma-separated. Wget needs to be built with libcares for this option to be available.

-t *number*

--tries=*number*

Set number of tries to *number*. Specify 0 or **inf** for infinite retrying. The default is to retry 20 times, with the exception of fatal errors like "connection refused" or "not found" (404), which are not retried.

-O *file*

--output-document=*file*

The documents will not be written to the appropriate files, but all will be concatenated together and written to *file*. If **-** is used as *file*, documents will be printed to standard output, disabling link conversion. (Use **./-** to print to a file literally named **-**.)

Use of **-O** is *not* intended to mean simply "use the name *file* instead of the one in the URL ;" rather, it is analogous to shell redirection: **wget -O file http://foo** is intended to work like **wget -O - http://foo > file**; *file* will be truncated immediately, and *all* downloaded content will be written there.

For this reason, **-N** (for timestamp-checking) is not supported in combination with **-O**: since *file* is always newly created, it will always have a very new timestamp. A warning will be issued if this combination is used.

Similarly, using **-r** or **-p** with **-O** may not work as you expect: Wget won't just download the first file to *file* and then download the rest to their normal names: *all* downloaded content will be placed in *file*. This was disabled in version 1.11, but has been reinstated (with a warning) in 1.11.2, as there are some cases where this behavior can actually have some use.

A combination with **-nc** is only accepted if the given output file does not exist.

Note that a combination with **-k** is only permitted when downloading a single document, as in that case it will just convert all relative URIs to external ones; **-k** makes no sense for multiple URIs when they're all being downloaded to a single file; **-k** can be used only when the output is a regular file.

-nc
--no-clobber

If a file is downloaded more than once in the same directory, Wget's behavior depends on a few options, including **-nc**. In certain cases, the local file will be *clobbered*, or overwritten, upon repeated download. In other cases it will be preserved.

When running Wget without **-N**, **-nc**, **-r**, or **-p**, downloading the same file in the same directory will result in the original copy of *file* being preserved and the second copy being named *file***.1**. If that file is downloaded yet again, the third copy will be named *file***.2**, and so on. (This is also the behavior with **-nd**, even if **-r** or **-p** are in effect.) When **-nc** is specified, this behavior is suppressed, and Wget will refuse to download newer copies of *file*. Therefore, `""no-clobber""` is actually a misnomer in this mode---it's not clobbering that's prevented (as the numeric suffixes were already preventing clobbering), but rather the multiple version saving that's prevented.

When running Wget with **-r** or **-p**, but without **-N**, **-nd**, or **-nc**, re-downloading a file will result in the new copy simply overwriting the old. Adding **-nc** will prevent this behavior, instead causing the original version to be preserved and any newer copies on the server to be ignored.

When running Wget with **-N**, with or without **-r** or **-p**, the decision as to whether or not to download a newer copy of a file depends on the local and remote timestamp and size of the file. **-nc** may not be specified at the same time as **-N**.

A combination with **-O**/**--output-document** is only accepted if the given output file does not exist.

Note that when **-nc** is specified, files with the suffixes **.html** or **.htm** will be loaded from the local disk and parsed as if they had been retrieved from the Web.

--backups=*backups*

Before (over)writing a file, back up an existing file by adding a **.1** suffix (**_1** on VMS) to the file name. Such backup files are rotated to **.2**, **.3**, and so on, up to *backups* (and lost beyond that).

--no-netrc

Do not try to obtain credentials from *.netrc* file. By default *.netrc* file is searched for credentials in case none have been passed on command line and authentication is required.

-c

--continue

Continue getting a partially-downloaded file. This is useful when you want to finish up a download started by a previous instance of Wget, or by another program. For instance:

```
wget -c ftp://sunsite.doc.ic.ac.uk/ls-lR.Z
```

If there is a file named *ls-lR.Z* in the current directory, Wget will assume that it is the first portion of the remote file, and will ask the server to continue the retrieval from an offset equal to the length of the local file.

Note that you don't need to specify this option if you just want the current invocation of Wget to retry downloading a file should the connection be lost midway through. This is the default behavior. **-c** only affects resumption of downloads started *prior* to this invocation of Wget, and whose local files are still sitting around.

Without **-c**, the previous example would just download the remote file to *ls-lR.Z.1*, leaving the truncated *ls-lR.Z* file alone.

If you use **-c** on a non-empty file, and the server does not support continued downloading, Wget will restart the download from scratch and overwrite the existing file entirely.

Beginning with Wget 1.7, if you use **-c** on a file which is of equal size as the one on the server, Wget will refuse to download the file and print an explanatory message. The same happens when the file is smaller on the server than locally (presumably because it was changed on the server since your last download attempt)---because "continuing" is not meaningful, no download occurs.

On the other side of the coin, while using **-c**, any file that's bigger on the server than locally will be considered an incomplete download and only "`(length(remote) - length(local))`" bytes will be downloaded and tacked onto the end of the local file. This behavior can be desirable in certain cases-

--for instance, you can use **wget -c** to download just the new portion that's been appended to a data collection or log file.

However, if the file is bigger on the server because it's been *changed*, as opposed to just *appended* to, you'll end up with a garbled file. Wget has no way of verifying that the local file is really a valid prefix of the remote file. You need to be especially careful of this when using **-c** in conjunction with **-r**, since every file will be considered as an "incomplete download" candidate.

Another instance where you'll get a garbled file if you try to use **-c** is if you have a lame HTTP proxy that inserts a "transfer interrupted" string into the local file. In the future a "rollback" option may be added to deal with this case.

Note that **-c** only works with FTP servers and with HTTP servers that support the `"Range"` header.

--start-pos= *OFFSET*

Start downloading at zero-based position *OFFSET* . Offset may be expressed in bytes, kilobytes with the 'k' suffix, or megabytes with the 'm' suffix, etc.

--start-pos has higher precedence over **--continue**. When **--start-pos** and**--continue** are both specified, wget will emit a warning then proceed as if**--continue** was absent.

Server support for continued download is required, otherwise **--start-pos**cannot help. See **-c** for details.

--progress=*type*

Select the type of the progress indicator you wish to use. Legal indicators are "dot" and "bar".

The "bar" indicator is used by default. It draws an ASCII progress bar graphics (a.k.a "thermometer" display) indicating the status of retrieval. If the output is not a TTY, the "dot" bar will be used by default.

Use **--progress=dot** to switch to the "dot" display. It traces the retrieval by printing dots on the screen, each dot representing a fixed amount of downloaded data.

The progress *type* can also take one or more parameters. The parameters vary based on the *type* selected. Parameters to *type* are passed by appending them to the type sperated by a colon (:) like this: **--progress=***type***:***parameter1***:***parameter2*.

When using the dotted retrieval, you may set the *style* by specifying the type as **dot:***style*. Different styles assign different meaning to one dot. With the `"default"` style each dot represents 1K, there are ten dots in a cluster and 50 dots in a line. The `"binary"` style has a more "computer"-like orientation---8K dots, 16-dots clusters and 48 dots per line (which makes for 384K lines). The `"mega"` style is suitable for downloading large files---each dot

represents 64K retrieved, there are eight dots in a cluster, and 48 dots on each line (so each line contains 3M). If `"mega"` is not enough then you can use the `"giga"` style---each dot represents 1M retrieved, there are eight dots in a cluster, and 32 dots on each line (so each line contains 32M).

With **--progress=bar**, there are currently two possible parameters, *force*and *noscroll*.

When the output is not a TTY, the progress bar always falls back to "dot", even if **--progress=bar** was passed to Wget during invocation. This behaviour can be overridden and the "bar" output forced by using the "force" parameter as **--progress=bar:force**.

By default, the **bar** style progress bar scroll the name of the file from left to right for the file being downloaded if the filename exceeds the maximum length allotted for its display. In certain cases, such as with **--progress=bar:force**, one may not want the scrolling filename in the progress bar. By passing the "noscroll" parameter, Wget can be forced to display as much of the filename as possible without scrolling through it.

Note that you can set the default style using the `"progress"` command in *.wgetrc*. That setting may be overridden from the command line. For example, to force the bar output without scrolling, use **--progress=bar:force:noscroll**.

--show-progress

Force wget to display the progress bar in any verbosity.

By default, wget only displays the progress bar in verbose mode. One may however, want wget to display the progress bar on screen in conjunction with any other verbosity modes like **--no-verbose** or **--quiet**. This is often a desired a property when invoking wget to download several small/large files. In such a case, wget could simply be invoked with this parameter to get a much cleaner output on the screen.

This option will also force the progress bar to be printed to *stderr* when used alongside the **--output-file** option.

-N

--timestamping

Turn on time-stamping.

--no-if-modified-since

Do not send If-Modified-Since header in **-N** mode. Send preliminaryHEAD request instead. This has only effect in **-N** mode.

--no-use-server-timestamps

Don’t set the local file’s timestamp by the one on the server.

By default, when a file is downloaded, its timestamps are set to match those from the remote file. This allows the use of **--timestamping** on subsequent invocations of wget. However, it is sometimes useful to base the local file’s timestamp on when it was actually downloaded; for that purpose, the **--no-use-server-timestamps** option has been provided.

-S
--server-response
Print the headers sent by HTTP servers and responses sent by FTP servers.
--spider
When invoked with this option, Wget will behave as a Web *spider*, which means that it will not download the pages, just check that they are there. For example, you can use Wget to check your bookmarks:

```
wget --spider --force-html -i bookmarks.html
```

This feature needs much more work for Wget to get close to the functionality of real web spiders.
-T seconds
--timeout=*seconds*
Set the network timeout to *seconds* seconds. This is equivalent to specifying **--dns-timeout**, **--connect-timeout**, and **--read-timeout**, all at the same time.

When interacting with the network, Wget can check for timeout and abort the operation if it takes too long. This prevents anomalies like hanging reads and infinite connects. The only timeout enabled by default is a 900-second read timeout. Setting a timeout to 0 disables it altogether. Unless you know what you are doing, it is best not to change the default timeout settings.

All timeout-related options accept decimal values, as well as subsecond values. For example, **0.1** seconds is a legal (though unwise) choice of timeout. Subsecond timeouts are useful for checking server response times or for testing network latency.
--dns-timeout=*seconds*
Set the DNS lookup timeout to *seconds* seconds. DNS lookups that don't complete within the specified time will fail. By default, there is no timeout on DNS lookups, other than that implemented by system libraries.
--connect-timeout=*seconds*
Set the connect timeout to *seconds* seconds. TCP connections that take longer to establish will be aborted. By default, there is no connect timeout, other than that implemented by system libraries.
--read-timeout=*seconds*
Set the read (and write) timeout to *seconds* seconds. The "time" of this timeout refers to *idle time*: if, at any point in the download, no data is received for more than the specified number of seconds, reading fails and the download is restarted. This option does not directly affect the duration of the entire download.

Of course, the remote server may choose to terminate the connection sooner than this option requires. The default read timeout is 900 seconds.
--limit-rate=*amount*
Limit the download speed to *amount* bytes per second. Amount may be expressed in bytes, kilobytes with the **k** suffix, or megabytes with the **m**suffix. For example, **--limit-rate=20k** will limit the retrieval rate to

20KB/s. This is useful when, for whatever reason, you don't want Wget to consume the entire available bandwidth.

This option allows the use of decimal numbers, usually in conjunction with power suffixes; for example, **--limit-rate=2.5k** is a legal value.

Note that Wget implements the limiting by sleeping the appropriate amount of time after a network read that took less time than specified by the rate. Eventually this strategy causes the TCP transfer to slow down to approximately the specified rate. However, it may take some time for this balance to be achieved, so don't be surprised if limiting the rate doesn't work well with very small files.

-w *seconds*
--wait=*seconds*

Wait the specified number of seconds between the retrievals. Use of this option is recommended, as it lightens the server load by making the requests less frequent. Instead of in seconds, the time can be specified in minutes using the `"m"` suffix, in hours using `"h"` suffix, or in days using `"d"` suffix.

Specifying a large value for this option is useful if the network or the destination host is down, so that Wget can wait long enough to reasonably expect the network error to be fixed before the retry. The waiting interval specified by this function is influenced by `"--random-wait"`, which see.

--waitretry=*seconds*

If you don't want Wget to wait between *every* retrieval, but only between retries of failed downloads, you can use this option. Wget will use *linear backoff*, waiting 1 second after the first failure on a given file, then waiting 2 seconds after the second failure on that file, up to the maximum number of *seconds* you specify.

By default, Wget will assume a value of 10 seconds.

--random-wait

Some web sites may perform log analysis to identify retrieval programs such as Wget by looking for statistically significant similarities in the time between requests. This option causes the time between requests to vary between 0.5 and 1.5 * *wait* seconds, where *wait* was specified using the **--wait** option, in order to mask Wget's presence from such analysis.

A 2001 article in a publication devoted to development on a popular consumer platform provided code to perform this analysis on the fly. Its author suggested blocking at the class C address level to ensure automated retrieval programs were blocked despite changing DHCP-supplied addresses.

The **--random-wait** option was inspired by this ill-advised recommendation to block many unrelated users from a web site due to the actions of one.

--no-proxy

Don't use proxies, even if the appropriate `*_proxy` environment variable is defined.

-Q *quota*
--quota=*quota*
Specify download quota for automatic retrievals. The value can be specified in bytes (default), kilobytes (with **k** suffix), or megabytes (with **m** suffix).

Note that quota will never affect downloading a single file. So if you specify **wget -Q10k https://example.com/ls-lR.gz**, all of the *ls-lR.gz* will be downloaded. The same goes even when several URLs are specified on the command-line. However, quota is respected when retrieving either recursively, or from an input file. Thus you may safely type **wget -Q2m -i sites**---download will be aborted when the quota is exceeded.

Setting quota to 0 or to **inf** unlimits the download quota.
--no-dns-cache
Turn off caching of DNS lookups. Normally, Wget remembers the IPaddresses it looked up from DNS so it doesn't have to repeatedly contact the DNS server for the same (typically small) set of hosts it retrieves from. This cache exists in memory only; a new Wget run will contact DNSagain.

However, it has been reported that in some situations it is not desirable to cache host names, even for the duration of a short-running application like Wget. With this option Wget issues a new DNS lookup (more precisely, a new call to `"gethostbyname"` or `"getaddrinfo"`) each time it makes a new connection. Please note that this option will *not* affect caching that might be performed by the resolving library or by an external caching layer, such as NSCD.

If you don't understand exactly what this option does, you probably won't need it.
--restrict-file-names=*modes*
Change which characters found in remote URLs must be escaped during generation of local filenames. Characters that are *restricted* by this option are escaped, i.e. replaced with **%HH**, where **HH** is the hexadecimal number that corresponds to the restricted character. This option may also be used to force all alphabetical cases to be either lower- or uppercase.

By default, Wget escapes the characters that are not valid or safe as part of file names on your operating system, as well as control characters that are typically unprintable. This option is useful for changing these defaults, perhaps because you are downloading to a non-native partition, or because you want to disable escaping of the control characters, or you want to further restrict characters to only those in the ASCII range of values.

The *modes* are a comma-separated set of text values. The acceptable values are **unix**, **windows**, **nocontrol**, **ascii**, **lowercase**, and **uppercase**. The values **unix** and **windows** are mutually exclusive (one will override the other), as are **lowercase** and **uppercase**. Those last are special cases, as they do not change the set of characters that would be escaped, but rather force local file paths to be converted either to lower- or uppercase.

When "unix" is specified, Wget escapes the character / and the control characters in the ranges 0--31 and 128--159. This is the default on Unix-like operating systems.

When "windows" is given, Wget escapes the characters \, |, /, **:**, **?**, **"**, *, <, >, and the control characters in the ranges 0--31 and 128--159. In addition to this, Wget in Windows mode uses + instead of **:** to separate host and port in local file names, and uses @ instead of **?** to separate the query portion of the file name from the rest. Therefore, a URL that would be saved as **www.xemacs.org:4300/search.pl?input=blah** in Unix mode would be saved as **www.xemacs.org+4300/search.pl@input=blah** in Windows mode. This mode is the default on Windows.

If you specify **nocontrol**, then the escaping of the control characters is also switched off. This option may make sense when you are downloading URLs whose names contain UTF-8 characters, on a system which can save and display filenames in UTF-8 (some possible byte values used in UTF-8 byte sequences fall in the range of values designated by Wget as "controls").

The **ascii** mode is used to specify that any bytes whose values are outside the range of ASCII characters (that is, greater than 127) shall be escaped. This can be useful when saving filenames whose encoding does not match the one used locally.

-4
--inet4-only
-6
--inet6-only

Force connecting to IPv4 or IPv6 addresses. With **--inet4-only** or **-4**, Wget will only connect to IPv4 hosts, ignoring AAAA records in DNS, and refusing to connect to IPv6 addresses specified in URLs. Conversely, with **--inet6-only** or **-6**, Wget will only connect to IPv6 hosts and ignore A records and IPv4 addresses.

Neither options should be needed normally. By default, an IPv6-aware Wget will use the address family specified by the host's DNS record. If theDNS responds with both IPv4 and IPv6 addresses, Wget will try them in sequence until it finds one it can connect to. (Also see `"--prefer-family"`option described below.)

These options can be used to deliberately force the use of IPv4 or IPv6 address families on dual family systems, usually to aid debugging or to deal with broken network configuration. Only one of **--inet6-only** and **--inet4-only** may be specified at the same time. Neither option is available in Wget compiled without IPv6 support.

--prefer-family=none/IPv4/IPv6

When given a choice of several addresses, connect to the addresses with specified address family first. The address order returned by DNS is used without change by default.

This avoids spurious errors and connect attempts when accessing hosts that resolve to both IPv6 and IPv4 addresses from IPv4 networks. For example, **www.kame.net** resolves to**2001:200:0:8002:203:47ff:fea5:3085** and to **203.178.141.194**. When the preferred family is `"IPv4"`, the IPv4 address is used first; when the preferred family is `"IPv6"`, the IPv6 address is used first; if the specified value is `"none"`, the address order returned by DNS is used without change.

Unlike **-4** and **-6**, this option doesn't inhibit access to any address family, it only changes the *order* in which the addresses are accessed. Also note that the reordering performed by this option is *stable*---it doesn't affect order of addresses of the same family. That is, the relative order of all IPv4 addresses and of all IPv6 addresses remains intact in all cases.

--retry-connrefused

Consider "connection refused" a transient error and try again. Normally Wget gives up on a URL when it is unable to connect to the site because failure to connect is taken as a sign that the server is not running at all and that retries would not help. This option is for mirroring unreliable sites whose servers tend to disappear for short periods of time.

--user=*user*

--password=*password*

Specify the username *user* and password *password* for both FTP and HTTPfile retrieval. These parameters can be overridden using the **--ftp-user**and **--ftp-password** options for FTP connections and the **--http-user** and **--http-password** options for HTTP connections.

--ask-password

Prompt for a password for each connection established. Cannot be specified when **--password** is being used, because they are mutually exclusive.

--use-askpass=*command*

Prompt for a user and password using the specified command. If no command is specified then the command in the environment variableWGET_ASKPASS is used. If WGET_ASKPASS is not set then the command in the environment variable SSH_ASKPASS is used.

You can set the default command for use-askpass in the *.wgetrc*. That setting may be overridden from the command line.

--no-iri

Turn off internationalized URI (IRI) support. Use **--iri** to turn it on. IRIsupport is activated by default.

You can set the default state of IRI support using the `"iri"` command in *.wgetrc*. That setting may be overridden from the command line.

--local-encoding=*encoding*

Force Wget to use *encoding* as the default system encoding. That affects how Wget converts URLs specified as arguments from locale to UTF-8 for IRI support.

Wget use the function `"nl_langinfo()"` and then the `"CHARSET"`environment variable to get the locale. If it fails, ASCII is used.

You can set the default local encoding using the `"local_encoding"`command in *.wgetrc*. That setting may be overridden from the command line.

--remote-encoding=*encoding*

Force Wget to use *encoding* as the default remote server encoding. That affects how Wget converts URIs found in files from remote encoding to UTF-8 during a recursive fetch. This options is only useful for IRI support, for the interpretation of non-ASCII characters.

For HTTP, remote encoding can be found in HTTP `"Content-Type"` header and in HTML `"Content-Type http-equiv"` meta tag.

You can set the default encoding using the `"remoteencoding"` command in.*wgetrc*. That setting may be overridden from the command line.

--unlink

Force Wget to unlink file instead of clobbering existing file. This option is useful for downloading to the directory with hardlinks.

Directory Options

-nd

--no-directories

Do not create a hierarchy of directories when retrieving recursively. With this option turned on, all files will get saved to the current directory, without clobbering (if a name shows up more than once, the filenames will get extensions **.n**).

-x

--force-directories

The opposite of **-nd**---create a hierarchy of directories, even if one would not have been created otherwise. E.g. **wget -x http://fly.srk.fer.hr/robots.txt** will save the downloaded file to *fly.srk.fer.hr/robots.txt*.

-

nH

--no-host-directories

Disable generation of host-prefixed directories. By default, invoking Wget with **-r http://fly.srk.fer.hr/** will create a structure of directories beginning with *fly.srk.fer.hr/*. This option disables such behavior.

--protocol-directories

Use the protocol name as a directory component of local file names. For example, with this option, **wget -r http://***host* will save to **http/***host***/...**rather than just to *host***/...**.

--cut-dirs=*number*

Ignore *number* directory components. This is useful for getting a fine-grained control over the directory where recursive retrieval will be saved.

Take, for example, the directory at **ftp://ftp.xemacs.org/pub/xemacs/**. If you retrieve it with **-r**, it will be saved locally under*ftp.xemacs.org/pub/xemacs/*. While the **-nH** option can remove the*ftp.xemacs.org/* part, you are still stuck with *pub/xemacs*. This is where **--cut-dirs** comes in handy; it makes Wget not "see" *number* remote directory components. Here are several examples of how **--cut-dirs**option works.

```
No options        -> ftp.xemacs.org/pub/xemacs/
-nH               -> pub/xemacs/
-nH --cut-dirs=1  -> xemacs/
-nH --cut-dirs=2  -> .
--cut-dirs=1      -> ftp.xemacs.org/xemacs/
...
```

If you just want to get rid of the directory structure, this option is similar to a combination of **-nd** and **-P**. However, unlike **-nd**, **--cut-dirs** does not lose with subdirectories---for instance, with **-nH --cut-dirs=1**, a *beta/*subdirectory will be placed to *xemacs/beta*, as one would expect.

-P *prefix*
--directory-prefix=*prefix*

Set directory prefix to *prefix*. The *directory prefix* is the directory where all other files and subdirectories will be saved to, i.e. the top of the retrieval tree. The default is **.** (the current directory).

HTTP Options

--default-page=*name*

Use *name* as the default file name when it isn't known (i.e., for URLs that end in a slash), instead of *index.html*.

-E
--adjust-extension

If a file of type **application/xhtml+xml** or **text/html** is downloaded and the URL does not end with the regexp **\.[Hh][Tt][Mm][Ll]?**, this option will cause the suffix **.html** to be appended to the local filename. This is useful, for instance, when you're mirroring a remote site that uses **.asp**pages, but you want the mirrored pages to be viewable on your stock Apache server. Another good use for this is when you're downloading CGI-generated materials. A URL like **http://site.com/article.cgi?25** will be saved as *article.cgi?25.html*.

Note that filenames changed in this way will be re-downloaded every time you re-mirror a site, because Wget can't tell that the local *X.html* file corresponds to remote URL *X* (since it doesn't yet know that the URLproduces output of type **text/html** or **application/xhtml+xml**.

As of version 1.12, Wget will also ensure that any downloaded files of type **text/css** end in the suffix **.css**, and the option was renamed from **--html-extension**, to better reflect its new behavior. The old option name is still acceptable, but should now be considered deprecated.

As of version 1.19.2, Wget will also ensure that any downloaded files with a `"Content-Encoding"` of **br**, **compress**, **deflate** or **gzip** end in the suffix **.br**, **.Z**, **.zlib** and **.gz** respectively.

At some point in the future, this option may well be expanded to include suffixes for other types of content, including content types that are not parsed by Wget.

--http-user=*user*
--http-password=*password*

Specify the username *user* and password *password* on an HTTP server. According to the type of the challenge, Wget will encode them using either the `"basic"` (insecure), the `"digest"`, or the Windows `"NTLM"`authentication scheme.

Another way to specify username and password is in the URL itself. Either method reveals your password to anyone who bothers to run `"ps"`. To prevent the passwords from being seen, use the **--use-askpass** or store them in *.wgetrc* or *.netrc*, and make sure to protect those files from other users with `"chmod"`. If the passwords are really important, do not leave them lying in those files either---edit the files and delete them after Wget has started the download.

--no-http-keep-alive

Turn off the "keep-alive" feature for HTTP downloads. Normally, Wget asks the server to keep the connection open so that, when you download more than one document from the same server, they get transferred over the same TCP connection. This saves time and at the same time reduces the load on the server.

This option is useful when, for some reason, persistent (keep-alive) connections don't work for you, for example due to a server bug or due to the inability of server-side scripts to cope with the connections.

--no-cache

Disable server-side cache. In this case, Wget will send the remote server appropriate directives (**Cache-Control: no-cache** and **Pragma: no-cache**) to get the file from the remote service, rather than returning the cached version. This is especially useful for retrieving and flushing out-of-date documents on proxy servers.

Caching is allowed by default.

--no-cookies

Disable the use of cookies. Cookies are a mechanism for maintaining server-side state. The server sends the client a cookie using the `"Set-Cookie"` header, and the client responds with the same cookie upon further requests. Since cookies allow the server owners to keep track of visitors and for sites to exchange this information, some consider them a breach of privacy. The default is to use cookies; however, *storing* cookies is not on by default.

--load-cookies *file*

Load cookies from *file* before the first HTTP retrieval. *file* is a textual file in the format originally used by Netscape's *cookies.txt* file.

You will typically use this option when mirroring sites that require that you be logged in to access some or all of their content. The login process typically works by the web server issuing an HTTP cookie upon receiving and verifying your credentials. The cookie is then resent by the browser when accessing that part of the site, and so proves your identity.

Mirroring such a site requires Wget to send the same cookies your browser sends when communicating with the site. This is achieved by **--load-**

cookies---simply point Wget to the location of the *cookies.txt* file, and it will send the same cookies your browser would send in the same situation. Different browsers keep textual cookie files in different locations:

"Netscape 4.x."

The cookies are in *~/.netscape/cookies.txt*.

"Mozilla and Netscape 6.x."

Mozilla's cookie file is also named *cookies.txt*, located somewhere under *~/.mozilla*, in the directory of your profile. The full path usually ends up looking somewhat like *~/.mozilla/default/some-weird-string/cookies.txt*.

"Internet Explorer."

You can produce a cookie file Wget can use by using the File menu, Import and Export, Export Cookies. This has been tested with Internet Explorer 5; it is not guaranteed to work with earlier versions.

"Other browsers."

If you are using a different browser to create your cookies, **--load-cookies** will only work if you can locate or produce a cookie file in the Netscape format that Wget expects.

If you cannot use **--load-cookies**, there might still be an alternative. If your browser supports a "cookie manager", you can use it to view the cookies used when accessing the site you're mirroring. Write down the name and value of the cookie, and manually instruct Wget to send those cookies, bypassing the "official" cookie support:

```
wget --no-cookies --header "Cookie: <name>=<value>"
```

--save-cookies *file*

Save cookies to *file* before exiting. This will not save cookies that have expired or that have no expiry time (so-called "session cookies"), but also see **--keep-session-cookies**.

--keep-session-cookies

When specified, causes **--save-cookies** to also save session cookies. Session cookies are normally not saved because they are meant to be kept in memory and forgotten when you exit the browser. Saving them is useful on sites that require you to log in or to visit the home page before you can access some pages. With this option, multiple Wget runs are considered a single browser session as far as the site is concerned.

Since the cookie file format does not normally carry session cookies, Wget marks them with an expiry timestamp of 0. Wget's **--load-cookies**recognizes those as session cookies, but it might confuse other browsers. Also note that cookies so loaded will be treated as other session cookies, which means that if you want **--save-cookies** to preserve them again, you must use **--keep-session-cookies** again.

--ignore-length

Unfortunately, some HTTP servers (CGI programs, to be more precise) send out bogus `"Content-Length"` headers, which makes Wget go wild, as it thinks not all the document was retrieved. You can spot this syndrome if Wget retries getting the same document again and again, each time claiming that the (otherwise normal) connection has closed on the very same byte.

With this option, Wget will ignore the "Content-Length" header---as if it never existed.

--header=*header-line*

Send *header-line* along with the rest of the headers in each HTTP request. The supplied header is sent as-is, which means it must contain name and value separated by colon, and must not contain newlines.

You may define more than one additional header by specifying **--header**more than once.

```
wget --header='Accept-Charset: iso-8859-2' \
     --header='Accept-Language: hr'        \
       http://fly.srk.fer.hr/
```

Specification of an empty string as the header value will clear all previous user-defined headers.

As of Wget 1.10, this option can be used to override headers otherwise generated automatically. This example instructs Wget to connect to localhost, but to specify **foo.bar** in the "Host" header:

```
wget --header="Host: foo.bar" http://localhost/
```

In versions of Wget prior to 1.10 such use of **--header** caused sending of duplicate headers.

--compression=*type*

Choose the type of compression to be used. Legal values are **auto**, **gzip**and **none**.

If **auto** or **gzip** are specified, Wget asks the server to compress the file using the gzip compression format. If the server compresses the file and responds with the "Content-Encoding" header field set appropriately, the file will be decompressed automatically.

If **none** is specified, wget will not ask the server to compress the file and will not decompress any server responses. This is the default.

Compression support is currently experimental. In case it is turned on, please report any bugs to "bug-wget@gnu.org".

--max-redirect=*number*

Specifies the maximum number of redirections to follow for a resource. The default is 20, which is usually far more than necessary. However, on those occasions where you want to allow more (or fewer), this is the option to use.

--proxy-user=*user*

--proxy-password=*password*

Specify the username *user* and password *password* for authentication on a proxy server. Wget will encode them using the "basic" authentication scheme.

Security considerations similar to those with **--http-password** pertain here as well.

--referer=*url*

Include ‘Referer: *url*’ header in HTTP request. Useful for retrieving documents with server-side processing that assume they are always being retrieved by interactive web browsers and only come out properly when Referer is set to one of the pages that point to them.

--save-headers

Save the headers sent by the HTTP server to the file, preceding the actual contents, with an empty line as the separator.

-U *agent-string*

--user-agent=*agent-string*

Identify as *agent-string* to the HTTP server.

The HTTP protocol allows the clients to identify themselves using a `"User-Agent"` header field. This enables distinguishing the WWWsoftware, usually for statistical purposes or for tracing of protocol violations. Wget normally identifies as **Wget/***version*, *version* being the current version number of Wget.

However, some sites have been known to impose the policy of tailoring the output according to the `"User-Agent"`-supplied information. While this is not such a bad idea in theory, it has been abused by servers denying information to clients other than (historically) Netscape or, more frequently, Microsoft Internet Explorer. This option allows you to change the `"User-Agent"` line issued by Wget. Use of this option is discouraged, unless you really know what you are doing.

Specifying empty user agent with **--user-agent=""** instructs Wget not to send the `"User-Agent"` header in HTTP requests.

--post-data=*string*

--post-file=*file*

Use POST as the method for all HTTP requests and send the specified data in the request body. **--post-data** sends *string* as data, whereas **--post-file**sends the contents of *file*. Other than that, they work in exactly the same way. In particular, they *both* expect content of the form`"key1=value1&key2=value2"`, with percent-encoding for special characters; the only difference is that one expects its content as a command-line parameter and the other accepts its content from a file. In particular, **--post-file** is *not* for transmitting files as form attachments: those must appear as `"key=value"` data (with appropriate percent-coding) just like everything else. Wget does not currently support `"multipart/form-data"`for transmitting POST data; only `"application/x-www-form-urlencoded"`. Only one of **--post-data** and **--post-file** should be specified.

Please note that wget does not require the content to be of the form`"key1=value1&key2=value2"`, and neither does it test for it. Wget will simply transmit whatever data is provided to it. Most servers however expect the POST data to be in the above format when processing HTMLForms.

When sending a POST request using the **--post-file** option, Wget treats the file as a binary file and will send every character in the POST request without

stripping trailing newline or formfeed characters. Any other control characters in the text will also be sent as-is in the POST request.

Please be aware that Wget needs to know the size of the POST data in advance. Therefore the argument to `"--post-file"` must be a regular file; specifying a FIFO or something like */dev/stdin* won't work. It's not quite clear how to work around this limitation inherent in HTTP/1.0. Although HTTP/1.1 introduces *chunked* transfer that doesn't require knowing the request length in advance, a client can't use chunked unless it knows it's talking to an HTTP/1.1 server. And it can't know that until it receives a response, which in turn requires the request to have been completed -- a chicken-and-egg problem.

Note: As of version 1.15 if Wget is redirected after the POST request is completed, its behaviour will depend on the response code returned by the server. In case of a 301 Moved Permanently, 302 Moved Temporarily or 307 Temporary Redirect, Wget will, in accordance with RFC2616,continue to send a POST request. In case a server wants the client to change the Request method upon redirection, it should send a 303 See Other response code.

This example shows how to log in to a server using POST and then proceed to download the desired pages, presumably only accessible to authorized users:

```
# Log in to the server.  This can be done only once.
wget --save-cookies cookies.txt \
     --post-data 'user=foo&password=bar' \
     http://example.com/auth.php
# Now grab the page or pages we care about.
wget --load-cookies cookies.txt \
     -p http://example.com/interesting/article.php
```

If the server is using session cookies to track user authentication, the above will not work because **--save-cookies** will not save them (and neither will browsers) and the *cookies.txt* file will be empty. In that case use **--keep-session-cookies** along with **--save-cookies** to force saving of session cookies.

--method=*HTTP-Method*

For the purpose of RESTful scripting, Wget allows sending of other HTTPMethods without the need to explicitly set them using **--header=Header-Line**. Wget will use whatever string is passed to it after **--method** as the HTTP Method to the server.

--body-data=*Data-String*

--body-file=*Data-File*

Must be set when additional data needs to be sent to the server along with the Method specified using **--method**. **--body-data** sends *string* as data, whereas **--body-file** sends the contents of *file*. Other than that, they work in exactly the same way.

Currently, **--body-file** is *not* for transmitting files as a whole. Wget does not currently support `"multipart/form-data"` for transmitting data;

only`"application/x-www-form-urlencoded"`. In the future, this may be changed so that wget sends the **--body-file** as a complete file instead of sending its contents to the server. Please be aware that Wget needs to know the contents of BODY Data in advance, and hence the argument to **--body-file** should be a regular file. See **--post-file** for a more detailed explanation. Only one of **--body-data** and **--body-file** should be specified.

If Wget is redirected after the request is completed, Wget will suspend the current method and send a GET request till the redirection is completed. This is true for all redirection response codes except 307 Temporary Redirect which is used to explicitly specify that the request method should *not* change. Another exception is when the method is set to `"POST"`, in which case the redirection rules specified under **--post-data**are followed.

--content-disposition

If this is set to on, experimental (not fully-functional) support for`"Content-Disposition"` headers is enabled. This can currently result in extra round-trips to the server for a `"HEAD"` request, and is known to suffer from a few bugs, which is why it is not currently enabled by default.

This option is useful for some file-downloading CGI programs that use `"Content-Disposition"` headers to describe what the name of a downloaded file should be.

When combined with **--metalink-over-http** and **--trust-server-names**, a**Content-Type: application/metalink4+xml** file is named using the `"Content-Disposition"` filename field, if available.

--content-on-error

If this is set to on, wget will not skip the content when the server responds with a http status code that indicates error.

--trust-server-names

If this is set, on a redirect, the local file name will be based on the redirection URL. By default the local file name is based on the original URL. When doing recursive retrieving this can be helpful because in many web sites redirected URLs correspond to an underlying file structure, while link URLs do not.

--auth-no-challenge

If this option is given, Wget will send Basic HTTP authentication information (plaintext username and password) for all requests, just like Wget 1.10.2 and prior did by default.

Use of this option is not recommended, and is intended only to support some few obscure servers, which never send HTTP authentication challenges, but accept unsolicited auth info, say, in addition to form-based authentication.

--retry-on-host-error

Consider host errors, such as "Temporary failure in name resolution", as non-fatal, transient errors.

--retry-on-http-error=*code[,code,...]*

Consider given HTTP response codes as non-fatal, transient errors. Supply a comma-separated list of 3-digit HTTP response codes as argument. Useful to work around special circumstances where retries are required, but the server

responds with an error code normally not retried by Wget. Such errors might be 503 (Service Unavailable) and 429 (Too Many Requests). Retries enabled by this option are performed subject to the normal retry timing and retry count limitations of Wget.

Using this option is intended to support special use cases only and is generally not recommended, as it can force retries even in cases where the server is actually trying to decrease its load. Please use wisely and only if you know what you are doing.

HTTPS (SSL/TLS) Options

To support encrypted HTTP (HTTPS) downloads, Wget must be compiled with an external SSL library. The current default is GnuTLS. In addition, Wget also supports HSTS (HTTP Strict Transport Security). If Wget is compiled without SSL support, none of these options are available.

--secure-protocol=*protocol*

Choose the secure protocol to be used. Legal values are **auto**, **SSLv2**,**SSLv3**, **TLSv1**, **TLSv1_1**, **TLSv1_2**, **TLSv1_3** and **PFS** . If **auto** is used, the SSL library is given the liberty of choosing the appropriate protocol automatically, which is achieved by sending a TLSv1 greeting. This is the default.

Specifying **SSLv2**, **SSLv3**, **TLSv1**, **TLSv1_1**, **TLSv1_2** or **TLSv1_3**forces the use of the corresponding protocol. This is useful when talking to old and buggy SSL server implementations that make it hard for the underlying SSL library to choose the correct protocol version. Fortunately, such servers are quite rare.

Specifying **PFS** enforces the use of the so-called Perfect Forward Security cipher suites. In short, PFS adds security by creating a one-time key for each SSL connection. It has a bit more CPU impact on client and server. We use known to be secure ciphers (e.g. no MD4) and the TLS protocol. This mode also explicitly excludes non-PFS key exchange methods, such as RSA.

--https-only

When in recursive mode, only HTTPS links are followed.

--ciphers

Set the cipher list string. Typically this string sets the cipher suites and other SSL/TLS options that the user wish should be used, in a set order of preference (GnuTLS calls it 'priority string'). This string will be fed verbatim to the SSL/TLS engine (OpenSSL or GnuTLS) and hence its format and syntax is dependent on that. Wget will not process or manipulate it in any way. Refer to the OpenSSL or GnuTLS documentation for more information.

--no-check-certificate

Don't check the server certificate against the available certificate authorities. Also don't require the URL host name to match the common name presented by the certificate.

As of Wget 1.10, the default is to verify the server's certificate against the recognized certificate authorities, breaking the SSL handshake and aborting

the download if the verification fails. Although this provides more secure downloads, it does break interoperability with some sites that worked with previous Wget versions, particularly those using self-signed, expired, or otherwise invalid certificates. This option forces an "insecure" mode of operation that turns the certificate verification errors into warnings and allows you to proceed.

If you encounter "certificate verification" errors or ones saying that "common name doesn't match requested host name", you can use this option to bypass the verification and proceed with the download. *Only use this option if you are otherwise convinced of the site's authenticity, or if you really don't care about the validity of its certificate.* It is almost always a bad idea not to check the certificates when transmitting confidential or important data. For self-signed/internal certificates, you should download the certificate and verify against that instead of forcing this insecure mode. If you are really sure of not desiring any certificate verification, you can specify --check-certificate=quiet to tell wget to not print any warning about invalid certificates, albeit in most cases this is the wrong thing to do.

--certificate=*file*
Use the client certificate stored in *file*. This is needed for servers that are configured to require certificates from the clients that connect to them. Normally a certificate is not required and this switch is optional.

--certificate-type=*type*
Specify the type of the client certificate. Legal values are **PEM** (assumed by default) and **DER** , also known as **ASN1** .

--private-key=*file*
Read the private key from *file*. This allows you to provide the private key in a file separate from the certificate.

--private-key-type=*type*
Specify the type of the private key. Accepted values are **PEM** (the default) and **DER** .

--ca-certificate=*file*
Use *file* as the file with the bundle of certificate authorities (" CA") to verify the peers. The certificates must be in PEM format.

Without this option Wget looks for CA certificates at the system-specified locations, chosen at OpenSSL installation time.

--ca-directory=*directory*
Specifies directory containing CA certificates in PEM format. Each file contains one CA certificate, and the file name is based on a hash value derived from the certificate. This is achieved by processing a certificate directory with the "`c_rehash`" utility supplied with OpenSSL. Using **--ca-directory** is more efficient than **--ca-certificate** when many certificates are installed because it allows Wget to fetch certificates on demand.

Without this option Wget looks for CA certificates at the system-specified locations, chosen at OpenSSL installation time.

--crl-file=*file*
Specifies a CRL file in *file*. This is needed for certificates that have been revocated by the CAs.

--pinnedpubkey=file/hashes

Tells wget to use the specified public key file (or hashes) to verify the peer. This can be a path to a file which contains a single public key inPEM or DER format, or any number of base64 encoded sha256 hashes preceded by "sha256//" and separated by ";"

When negotiating a TLS or SSL connection, the server sends a certificate indicating its identity. A public key is extracted from this certificate and if it does not exactly match the public key(s) provided to this option, wget will abort the connection before sending or receiving any data.

--random-file=*file*

[OpenSSL and LibreSSL only] Use *file* as the source of random data for seeding the pseudo-random number generator on systems without*/dev/urandom*.

On such systems the SSL library needs an external source of randomness to initialize. Randomness may be provided by EGD (see **--egd-file** below) or read from an external source specified by the user. If this option is not specified, Wget looks for random data in `$RANDFILE` or, if that is unset, in*$HOME/.rnd*.

If you're getting the "Could not seed OpenSSL PRNG ; disabling SSL."error, you should provide random data using some of the methods described above.

--egd-file=*file*

[OpenSSL only] Use *file* as the EGD socket. EGD stands for *Entropy Gathering Daemon*, a user-space program that collects data from various unpredictable system sources and makes it available to other programs that might need it. Encryption software, such as the SSL library, needs sources of non-repeating randomness to seed the random number generator used to produce cryptographically strong keys.

OpenSSL allows the user to specify his own source of entropy using the`"RAND_FILE"` environment variable. If this variable is unset, or if the specified file does not produce enough randomness, OpenSSL will read random data from EGD socket specified using this option.

If this option is not specified (and the equivalent startup command is not used), EGD is never contacted. EGD is not needed on modern Unix systems that support */dev/urandom*.

--no-hsts

Wget supports HSTS (HTTP Strict Transport Security, RFC 6797) by default. Use **--no-hsts** to make Wget act as a non-HSTS-compliant UA.As a consequence, Wget would ignore all the `"Strict-Transport-Security"` headers, and would not enforce any existing HSTS policy.

--hsts-file=*file*

By default, Wget stores its HSTS database in *~/.wget-hsts*. You can use **--hsts-file** to override this. Wget will use the supplied file as the HSTSdatabase. Such file must conform to the correct HSTS database

format used by Wget. If Wget cannot parse the provided file, the behaviour is unspecified.

The Wget's HSTS database is a plain text file. Each line contains an HSTSentry (ie. a site that has issued a `"Strict-Transport-Security"` header and that therefore has specified a concrete HSTS policy to be applied). Lines starting with a dash (`"#"`) are ignored by Wget. Please note that in spite of this convenient human-readability hand-hacking the HSTS database is generally not a good idea.

An HSTS entry line consists of several fields separated by one or more whitespace:

```
"<hostname> SP [<port>] SP <include subdomains> SP <created> SP
<max-age>"
```

The *hostname* and *port* fields indicate the hostname and port to which the given HSTS policy applies. The *port* field may be zero, and it will, in most of the cases. That means that the port number will not be taken into account when deciding whether such HSTS policy should be applied on a given request (only the hostname will be evaluated). When *port* is different to zero, both the target hostname and the port will be evaluated and the HSTS policy will only be applied if both of them match. This feature has been included for testing/development purposes only. The Wget testsuite (in *testenv/*) creates HSTS databases with explicit ports with the purpose of ensuring Wget's correct behaviour. Applying HSTS policies to ports other than the default ones is discouraged by RFC 6797 (see Appendix B "Differences between HSTS Policy and Same-Origin Policy"). Thus, this functionality should not be used in production environments and *port* will typically be zero. The last three fields do what they are expected to. The field *include_subdomains* can either be `1` or `0`and it signals whether the subdomains of the target domain should be part of the given HSTS policy as well. The *created* and *max-age* fields hold the timestamp values of when such entry was created (first seen by Wget) and the HSTS-defined value 'max-age', which states how long should that HSTS policy remain active, measured in seconds elapsed since the timestamp stored in *created*. Once that time has passed, that HSTS policy will no longer be valid and will eventually be removed from the database.

If you supply your own HSTS database via **--hsts-file**, be aware that Wget may modify the provided file if any change occurs between the HSTSpolicies requested by the remote servers and those in the file. When Wget exists, it effectively updates the HSTS database by rewriting the database file with the new entries.

If the supplied file does not exist, Wget will create one. This file will contain the new HSTS entries. If no HSTS entries were generated (no`"Strict-Transport-Security"` headers were sent by any of the servers) then no file will be created, not even an empty one. This behaviour applies to the default

database file (*~/.wget-hsts*) as well: it will not be created until some server enforces an HSTS policy.

Care is taken not to override possible changes made by other Wget processes at the same time over the HSTS database. Before dumping the updated HSTS entries on the file, Wget will re-read it and merge the changes.

Using a custom HSTS database and/or modifying an existing one is discouraged. For more information about the potential security threats arose from such practice, see section 14 "Security Considerations" of RFC 6797, specially section 14.9 "Creative Manipulation of HSTS Policy Store".

--warc-file=*file*
Use *file* as the destination WARC file.

--warc-header=*string*
Use *string* into as the warcinfo record.

--warc-max-size=*size*
Set the maximum size of the WARC files to *size*.

--warc-cdx
Write CDX index files.

--warc-dedup=*file*
Do not store records listed in this CDX file.

--no-warc-compression
Do not compress WARC files with GZIP.

--no-warc-digests
Do not calculate SHA1 digests.

--no-warc-keep-log
Do not store the log file in a WARC record.

--warc-tempdir=*dir*
Specify the location for temporary files created by the WARC writer.

FTP Options

--ftp-user=*user*
--ftp-password=*password*
Specify the username *user* and password *password* on an FTP server. Without this, or the corresponding startup option, the password defaults to **-wget@**, normally used for anonymous FTP.

Another way to specify username and password is in the URL itself. Either method reveals your password to anyone who bothers to run `"ps"`. To prevent the passwords from being seen, store them in *.wgetrc* or *.netrc*, and make sure to protect those files from other users with `"chmod"`. If the passwords are really important, do not leave them lying in those files either--edit the files and delete them after Wget has started the download.

--no-remove-listing
Don't remove the temporary *.listing* files generated by FTP retrievals. Normally, these files contain the raw directory listings received from FTPservers. Not removing them can be useful for debugging purposes, or when you want to be able to easily check on the contents of remote server directories (e.g. to verify that a mirror you're running is complete).

Note that even though Wget writes to a known filename for this file, this is not a security hole in the scenario of a user making *.listing* a symbolic link to */etc/passwd* or something and asking `"root"` to run Wget in his or her directory. Depending on the options used, either Wget will refuse to write to *.listing*, making the globbing/recursion/time-stamping operation fail, or the symbolic link will be deleted and replaced with the actual *.listing* file, or the listing will be written to a *.listing.number* file.

Even though this situation isn't a problem, though, `"root"` should never run Wget in a non-trusted user's directory. A user could do something as simple as linking *index.html* to */etc/passwd* and asking `"root"` to run Wget with **-N** or **-r** so the file will be overwritten.

--no-glob

Turn off FTP globbing. Globbing refers to the use of shell-like special characters (*wildcards*), like *****, **?**, **[** and **]** to retrieve more than one file from the same directory at once, like:

```
wget ftp://gnjilux.srk.fer.hr/*.msg
```

By default, globbing will be turned on if the URL contains a globbing character. This option may be used to turn globbing on or off permanently.

You may have to quote the URL to protect it from being expanded by your shell. Globbing makes Wget look for a directory listing, which is system-specific. This is why it currently works only with Unix FTP servers (and the ones emulating Unix `"ls"` output).

--no-passive-ftp

Disable the use of the *passive* FTP transfer mode. Passive FTP mandates that the client connect to the server to establish the data connection rather than the other way around.

If the machine is connected to the Internet directly, both passive and active FTP should work equally well. Behind most firewall and NATconfigurations passive FTP has a better chance of working. However, in some rare firewall configurations, active FTP actually works when passiveFTP doesn't. If you suspect this to be the case, use this option, or set`"passive_ftp=off"` in your init file.

--preserve-permissions

Preserve remote file permissions instead of permissions set by umask.

--retr-symlinks

By default, when retrieving FTP directories recursively and a symbolic link is encountered, the symbolic link is traversed and the pointed-to files are retrieved. Currently, Wget does not traverse symbolic links to directories to download them recursively, though this feature may be added in the future.

When **--retr-symlinks=no** is specified, the linked-to file is not downloaded. Instead, a matching symbolic link is created on the local filesystem. The pointed-to file will not be retrieved unless this recursive retrieval would have encountered it separately and downloaded it anyway. This option poses a security risk where a malicious FTP Server may cause Wget to write to

files outside of the intended directories through a specially crafted .LISTING file.

Note that when retrieving a file (not a directory) because it was specified on the command-line, rather than because it was recursed to, this option has no effect. Symbolic links are always traversed in this case.

FTPS Options

--ftps-implicit

This option tells Wget to use FTPS implicitly. Implicit FTPS consists of initializing SSL/TLS from the very beginning of the control connection. This option does not send an `"AUTH TLS"` command: it assumes the server speaks FTPS and directly starts an SSL/TLS connection. If the attempt is successful, the session continues just like regular FTPS (`"PBSZ"` and `"PROT"`are sent, etc.). Implicit FTPS is no longer a requirement for FTPSimplementations, and thus many servers may not support it. If **--ftps-implicit** is passed and no explicit port number specified, the default port for implicit FTPS, 990, will be used, instead of the default port for the "normal" (explicit) FTPS which is the same as that of FTP, 21.

--no-ftps-resume-ssl

Do not resume the SSL/TLS session in the data channel. When starting a data connection, Wget tries to resume the SSL/TLS session previously started in the control connection. SSL/TLS session resumption avoids performing an entirely new handshake by reusing the SSL/TLS parameters of a previous session. Typically, the FTPS servers want it that way, so Wget does this by default. Under rare circumstances however, one might want to start an entirely new SSL/TLS session in every data connection. This is what **--no-ftps-resume-ssl** is for.

--ftps-clear-data-connection

All the data connections will be in plain text. Only the control connection will be under SSL/TLS. Wget will send a `"PROT C"` command to achieve this, which must be approved by the server.

--ftps-fallback-to-ftp

Fall back to FTP if FTPS is not supported by the target server. For security reasons, this option is not asserted by default. The default behaviour is to exit with an error. If a server does not successfully reply to the initial`"AUTH TLS"` command, or in the case of implicit FTPS, if the initial SSL/TLSconnection attempt is rejected, it is considered that such server does not support FTPS.

Recursive Retrieval Options

-r

--recursive

Turn on recursive retrieving. The default maximum depth is 5.

-l *depth*

--level=*depth*

Specify recursion maximum depth level *depth*.

--delete-after

This option tells Wget to delete every single file it downloads, *after*having done so. It is useful for pre-fetching popular pages through a proxy, e.g.:

```
wget -r -nd --delete-after http://whatever.com/~popular/page/
```

The **-r** option is to retrieve recursively, and **-nd** to not create directories.

Note that **--delete-after** deletes files on the local machine. It does not issue the **DELE** command to remote FTP sites, for instance. Also note that when **--delete-after** is specified, **--convert-links** is ignored, so **.orig** files are simply not created in the first place.

-k
--convert-links

After the download is complete, convert the links in the document to make them suitable for local viewing. This affects not only the visible hyperlinks, but any part of the document that links to external content, such as embedded images, links to style sheets, hyperlinks to non-HTML content, etc.

Each link will be changed in one of the two ways:

- The links to files that have been downloaded by Wget will be changed to refer to the file they point to as a relative link.

  Example: if the downloaded file */foo/doc.html* links to */bar/img.gif*, also downloaded, then the link in *doc.html* will be modified to point to **../bar/img.gif**. This kind of transformation works reliably for arbitrary combinations of directories.

- The links to files that have not been downloaded by Wget will be changed to include host name and absolute path of the location they point to.

  Example: if the downloaded file */foo/doc.html* links to */bar/img.gif*(or to *../bar/img.gif*), then the link in *doc.html* will be modified to point to *http://hostname/bar/img.gif*.

Because of this, local browsing works reliably: if a linked file was downloaded, the link will refer to its local name; if it was not downloaded, the link will refer to its full Internet address rather than presenting a broken link. The fact that the former links are converted to relative links ensures that you can move the downloaded hierarchy to another directory.

Note that only at the end of the download can Wget know which links have been downloaded. Because of that, the work done by **-k** will be performed at the end of all the downloads.

--convert-file-only

This option converts only the filename part of the URLs, leaving the rest of the URLs untouched. This filename part is sometimes referred to as the "basename", although we avoid that term here in order not to cause confusion.

It works particularly well in conjunction with **--adjust-extension**, although this coupling is not enforced. It proves useful to populate Internet caches with files downloaded from different hosts.

Example: if some link points to *//foo.com/bar.cgi?xyz* with **--adjust-extension** asserted and its local destination is intended to be*./foo.com/bar.cgi?xyz.css*, then the link would be converted to *//foo.com/bar.cgi?xyz.css*. Note that only the filename part has been modified. The rest of the URL has been left untouched, including the net path ("//") which would otherwise be processed by Wget and converted to the effective scheme (ie. "http://").

-K
--backup-converted

When converting a file, back up the original version with a **.orig** suffix. Affects the behavior of **-N**.

-m
--mirror

Turn on options suitable for mirroring. This option turns on recursion and time-stamping, sets infinite recursion depth and keeps FTP directory listings. It is currently equivalent to **-r -N -l inf --no-remove-listing**.

-p
--page-requisites

This option causes Wget to download all the files that are necessary to properly display a given HTML page. This includes such things as inlined images, sounds, and referenced stylesheets.

Ordinarily, when downloading a single HTML page, any requisite documents that may be needed to display it properly are not downloaded. Using **-r** together with **-l** can help, but since Wget does not ordinarily distinguish between external and inlined documents, one is generally left with "leaf documents" that are missing their requisites.

For instance, say document *1.html* contains an "<IMG>" tag referencing*1.gif* and an "<A>" tag pointing to external document *2.html*. Say that*2.html* is similar but that its image is *2.gif* and it links to *3.html*. Say this continues up to some arbitrarily high number.

If one executes the command:

```
wget -r -l 2 http://<site>/1.html
```

then *1.html*, *1.gif*, *2.html*, *2.gif*, and *3.html* will be downloaded. As you can see, *3.html* is without its requisite *3.gif* because Wget is simply counting the number of hops (up to 2) away from *1.html* in order to determine where to stop the recursion. However, with this command:

```
wget -r -l 2 -p http://<site>/1.html
```

all the above files *and 3.html*'s requisite *3.gif* will be downloaded. Similarly,

```
wget -r -l 1 -p http://<site>/1.html
```

will cause *1.html*, *1.gif*, *2.html*, and *2.gif* to be downloaded. One might think that:

```
wget -r -l 0 -p http://<site>/1.html
```

would download just *1.html* and *1.gif*, but unfortunately this is not the case, because **-l 0** is equivalent to **-l inf**---that is, infinite recursion. To download a single HTML page (or a handful of them, all specified on the command-line or in a **-i** URL input file) and its (or their) requisites, simply leave off **-r** and **-l**:

```
wget -p http://<site>/1.html
```

Note that Wget will behave as if **-r** had been specified, but only that single page and its requisites will be downloaded. Links from that page to external documents will not be followed. Actually, to download a single page and all its requisites (even if they exist on separate websites), and make sure the lot displays properly locally, this author likes to use a few options in addition to **-p**:

```
wget -E -H -k -K -p http://<site>/<document>
```

To finish off this topic, it's worth knowing that Wget's idea of an external document link is any URL specified in an `"<A>"` tag, an `"<AREA>"` tag, or a `"<LINK>"` tag other than `"<LINK REL="stylesheet">"`.

--strict-comments

Turn on strict parsing of HTML comments. The default is to terminate comments at the first occurrence of **-->**.

According to specifications, HTML comments are expressed as SGML*declarations*. Declaration is special markup that begins with **<!** and ends with >, such as **<!DOCTYPE ...>**, that may contain comments between a pair of **--** delimiters. HTML comments are "empty declarations", SGMLdeclarations without any non-comment text. Therefore, **<!--foo-->** is a valid comment, and so is **<!--one-- --two-->**, but **<!--1--2-->** is not.

On the other hand, most HTML writers don't perceive comments as anything other than text delimited with **<!--** and **-->**, which is not quite the same. For example, something like **<!---------->** works as a valid comment as long as the number of dashes is a multiple of four (!). If not, the comment technically lasts until the next **--**, which may be at the other end of the document. Because of this, many popular browsers completely ignore the specification and implement what users have come to expect: comments delimited with **<!--** and **-->**.

Until version 1.9, Wget interpreted comments strictly, which resulted in missing links in many web pages that displayed fine in browsers, but had the misfortune of containing non-compliant comments. Beginning with version 1.9, Wget has joined the ranks of clients that implements "naive" comments, terminating each comment at the first occurrence of **-->**.

If, for whatever reason, you want strict comment parsing, use this option to turn it on.

Recursive Accept/Reject Options

-A *acclist* **--accept** *acclist*

-R *rejlist* **--reject** *rejlist*

Specify comma-separated lists of file name suffixes or patterns to accept or reject. Note that if any of the wildcard characters, *****, **?**, **[** or **]**, appear in an element of *acclist* or *rejlist*, it will be treated as a pattern, rather than a suffix. In this case, you have to enclose the pattern into quotes to prevent your shell from expanding it, like in **-A "*.mp3"** or **-A '*.mp3'**.

--accept-regex *urlregex*

--reject-regex *urlregex*

Specify a regular expression to accept or reject the complete URL.

--regex-type *regextype*

Specify the regular expression type. Possible types are **posix** or **pcre**. Note that to be able to use **pcre** type, wget has to be compiled with libpcre support.

-D *domain-list*

--domains=*domain-list*

Set domains to be followed. *domain-list* is a comma-separated list of domains. Note that it does *not* turn on **-H**.

--exclude-domains *domain-list*

Specify the domains that are *not* to be followed.

--follow-ftp

Follow FTP links from HTML documents. Without this option, Wget will ignore all the FTP links.

--follow-tags=*list*

Wget has an internal table of HTML tag / attribute pairs that it considers when looking for linked documents during a recursive retrieval. If a user wants only a subset of those tags to be considered, however, he or she should be specify such tags in a comma-separated *list* with this option.

--ignore-tags=*list*

This is the opposite of the **--follow-tags** option. To skip certain HTML tags when recursively looking for documents to download, specify them in a comma-separated *list*.

In the past, this option was the best bet for downloading a single page and its requisites, using a command-line like:

```
wget --ignore-tags=a,area -H -k -K -r http://<site>/<document>
```

However, the author of this option came across a page with tags like `"<LINK REL="home" HREF="/">"` and came to the realization that specifying tags to ignore was not enough. One can't just tell Wget to ignore `"<LINK>"`, because then stylesheets will not be downloaded. Now the best bet for downloading a single page and its requisites is the dedicated **--page-requisites** option.

--ignore-case

Ignore case when matching files and directories. This influences the behavior of -R, -A, -I, and -X options, as well as globbing implemented when downloading from FTP sites. For example, with this option, **-A "*.txt"** will match **file1.txt**, but also **file2.TXT**, **file3.TxT**, and so on. The quotes in the example are to prevent the shell from expanding the pattern.

-H
--span-hosts
Enable spanning across hosts when doing recursive retrieving.
-L
--relative
Follow relative links only. Useful for retrieving a specific home page without any distractions, not even those from the same hosts.
-I *list*
--include-directories=*list*
Specify a comma-separated list of directories you wish to follow when downloading. Elements of *list* may contain wildcards.
-X *list*
--exclude-directories=*list*
Specify a comma-separated list of directories you wish to exclude from download. Elements of *list* may contain wildcards.
-np
--no-parent
Do not ever ascend to the parent directory when retrieving recursively. This is a useful option, since it guarantees that only the files *below* a certain hierarchy will be downloaded.

ENVIRONMENT

Wget supports proxies for both HTTP and FTP retrievals. The standard way to specify proxy location, which Wget recognizes, is using the following environment variables:
http_proxy
https_proxy
If set, the **http_proxy** and **https_proxy** variables should contain the URLs of the proxies for HTTP and HTTPS connections respectively.
ftp_proxy
This variable should contain the URL of the proxy for FTP connections. It is quite common that **http_proxy** and **ftp_proxy** are set to the same URL.
no_proxy
This variable should contain a comma-separated list of domain extensions proxy should *not* be used for. For instance, if the value of **no_proxy** is **.mit.edu**, proxy will not be used to retrieve documents from MIT.

EXIT STATUS

Wget may return one of several error codes if it encounters problems.
0 No problems occurred.
1 Generic error code.
2 Parse error---for instance, when parsing command-line options, the **.wgetrc** or **.netrc**...
3 File I/O error.
4 Network failure.

5 SSL verification failure.
6 Username/password authentication failure.
7 Protocol errors.
8 Server issued an error response.

With the exceptions of 0 and 1, the lower-numbered exit codes take precedence over higher-numbered ones, when multiple types of errors are encountered.

In versions of Wget prior to 1.12, Wget's exit status tended to be unhelpful and inconsistent. Recursive downloads would virtually always return 0 (success), regardless of any issues encountered, and non-recursive fetches only returned the status corresponding to the most recently-attempted download.

FILES

/usr/local/etc/wgetrc
Default location of the *global* startup file.
.wgetrc
User startup file.

BUGS

You are welcome to submit bug reports via the GNU Wget bug tracker (see <**https://savannah.gnu.org/bugs/?func=additem&group=wget**>) or to our mailing list <**bug-wget@gnu.org**>.

Visit <**https://lists.gnu.org/mailman/listinfo/bug-wget**> to get more info (how to subscribe, list archives, ...).

Before actually submitting a bug report, please try to follow a few simple guidelines.

1. Please try to ascertain that the behavior you see really is a bug. If Wget crashes, it's a bug. If Wget does not behave as documented, it's a bug. If things work strange, but you are not sure about the way they are supposed to work, it might well be a bug, but you might want to double-check the documentation and the mailing lists.
2. Try to repeat the bug in as simple circumstances as possible. E.g. if Wget crashes while downloading **wget -rl0 -kKE -t5 --no-proxy http://example.com -o /tmp/log**, you should try to see if the crash is repeatable, and if will occur with a simpler set of options. You might even try to start the download at the page where the crash occurred to see if that page somehow triggered the crash.

   Also, while I will probably be interested to know the contents of your *.wgetrc* file, just dumping it into the debug message is probably a bad idea. Instead, you should first try to see if the bug repeats with *.wgetrc* moved out of the way. Only if it turns out that *.wgetrc* settings affect the bug, mail me the relevant parts of the file.

3. Please start Wget with **-d** option and send us the resulting output (or relevant parts thereof). If Wget was compiled without debug support, recompile it---it is *much* easier to trace bugs with debug support on.

   Note: please make sure to remove any potentially sensitive information from the debug log before sending it to the bug address. The `"-d"` won't go out of its way to collect sensitive information, but the log *will* contain a fairly complete transcript of Wget's communication with the server, which may include passwords and pieces of downloaded data. Since the bug address is publicly archived, you may assume that all bug reports are visible to the public.

4. If Wget has crashed, try to run it in a debugger, e.g. `"gdb `which wget` core"` and type `"where"` to get the backtrace. This may not work if the system administrator has disabled core files, but it is safe to try.

SEE ALSO

This is **not** the complete manual for GNU Wget. For more complete information, including more detailed explanations of some of the options, and a number of commands available for use with *.wgetrc* files and the **-e** option, see the GNU Info entry for *wget*.