NAME

libcurl − client-side URL transfers

DESCRIPTION

This is an overview on how to use libcurl in your C programs. There are specific man pages for each function mentioned in here. There's also the libcurl-the-guide document for a complete tutorial to programming with libcurl.

There are a dozen custom bindings that bring libcurl access to your favourite language. Look elsewhere for documentation on those.

All applications that use libcurl should call *curl_global_init()* exactly once before any libcurl function can be used. After all usage of libcurl is complete, it **must** call *curl_global_cleanup()*. In between those two calls, you can use libcurl as described below.

When using libcurl's "easy" interface you init your session and get a handle, which you use as input to the easy interface functions you use. Use *curl_easy_init()* to get the handle. There is also the so called "multi" interface, try the *libcurl-multi(3)* man page for an overview of that.

You continue by setting all the options you want in the upcoming transfer, most important among them is the URL itself (you can't transfer anything without a specified URL as you may have figured out yourself). You might want to set some callbacks as well that will be called from the library when data is available etc. *curl_easy_setopt()* is there for this.

When all is setup, you tell libcurl to perform the transfer using *curl_easy_perform()*. It will then do the entire operation and won't return until it is done (successfully or not).

After the transfer has been made, you can set new options and make another transfer, or if you're done, cleanup the session by calling *curl_easy_cleanup()*. If you want persistant connections, you don't cleanup immediately, but instead run ahead and perform other transfers using the same handle. See the chapter below for Persistant Connections.

There is also a series of other helpful functions to use. They are:

curl_version()
displays the libcurl version

curl_getdate()
converts a date string to time_t

curl_getenv()
portable environment variable reader

curl_easy_getinfo()
get information about a performed transfer

curl_formadd()
helps building a HTTP form POST

curl_formfree()
free a list built with curl_formparse()/curl_formadd()

curl_slist_append()
builds a linked list

curl_slist_free_all()
frees a whole curl_slist

curl_mprintf()
portable printf() functions

curl_strequal()
portable case insensitive string comparisons

LINKING WITH LIBCURL

On unix-like machines, there's a tool named curl-config that gets installed with the rest of the curl stuff when 'make install' is performed.

curl-config is added to make it easier for applications to link with libcurl and developers to learn about libcurl and how to use it.

Run 'curl-config --libs' to get the (additional) linker options you need to link with the particular version of libcurl you've installed.

For details, see the curl-config.1 man page.

LIBCURL SYMBOL NAMES

All public functions in the libcurl interface are prefixed with 'curl_' (with a lowercase c). You can find other functions in the library source code, but other prefixes indicate the functions are private and may change without further notice in the next release.

Only use documented functions and functionality!

PORTABILITY

libcurl works **exactly** the same, on any of the platforms it compiles and builds on.

THREADS

Never ever call curl-functions simultaneously using the same handle from several threads. libcurl is thread-safe and can be used in any number of threads, but you must use separate curl handles if you want to use libcurl in more than one thread simultaneously.

PERSISTANT CONNECTIONS

Persistent connections means that libcurl can re-use the same connection for several transfers, if the conditions are right.

libcurl will *always* attempt to use persistent connections. Whenever you use curl_easy_perform(), libcurl will attempt to use an existing connection to do the transfer, and if none exists it'll open a new one that will be subject for re-use on a possible following call to curl_easy_perform().

To allow libcurl to take full advantage of persistent connections, you should do as many of your file transfers as possible using the same curl handle. When you call curl_easy_cleanup(), all the possibly open connections held by libcurl will be closed and forgotten.

Note that the options set with curl_easy_setopt() will be used in on every repeat curl_easy_perform() call