NAME

libcurl – client-side URL transfers

DESCRIPTION

This is an short overview on how to use libcurl in your C programs. There are specific man pages for each function mentioned in here. There are also the *libcurl-easy(3)* man page, the *libcurl-multi(3)* man page, the *libcurl-share(3)* man page and the *libcurl-tutorial(3)* man page for in-depth understanding on how to program with libcurl.

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

All applications that use libcurl should call *curl_global_init(3)* exactly once before any libcurl function can be used. After all usage of libcurl is complete, it **must** call *curl_global_cleanup(3)*. In between those two calls, you can use libcurl as described below.

To transfer files, you always set up an "easy handle" using *curl_easy_init(3)*, but when you want the file(s) transferred you have the option of using the "easy" interface, or the "multi" interface.

The easy interface is a synchronous interface with which you call *curl_easy_perform(3)* and let it perform the transfer. When it is completed, the function return and you can continue. More details are found in the *libcurl-easy(3)* man page.

The multi interface on the other hand is an asynchronous interface, that you call and that performs only a little piece of the transfer on each invoke. It is perfect if you want to do things while the transfer is in progress, or similar. The multi interface allows you to select() on libcurl action, and even to easily download multiple files simultaneously using a single thread. See further deails in the *libcurl-multi(3)* man page.

You can have multiple easy handles share certain data, even if they are used in different threads. This magic is setup using the share interface, as described in the *libcurl-share(3)* man page.

There is also a series of other helpful functions to use, including these:

curl_version_info()
: gets detailed libcurl (and other used libraries) version info

curl_getdate()
: converts a date string to time_t

curl_easy_getinfo()
: get information about a performed transfer

curl_formadd()
: helps building an HTTP form POST

curl_formfree()
: free a list built with *curl_formadd(3)*

curl_slist_append()
: builds a linked list

curl_slist_free_all()
: frees a whole curl_slist

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. See the *curl-config(1)* man page for further details.

Unix-like operating system that ship libcurl as part of their distributions often don't provide the curl-config tool, but simply install the library and headers in the common path for this purpose.

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 that 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.

PERSISTENT 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(3)* or *curl_multi_perform(3)*, 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(3)* or *curl_multi_perform(3)*.

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(3)*, all the possibly open connections held by libcurl will be closed and forgotten.

Note that the options set with *curl_easy_setopt(3)* will be used in on every repeated *curl_easy_perform(3)* call.