NAME

curl_global_cleanup - global libcurl cleanup

SYNOPSIS

#include <curl/curl.h>

void curl_global_cleanup(void);

DESCRIPTION

This function releases resources acquired by *curl_global_init(3)*.

You should call *curl_global_cleanup(3)* once for each call you make to *curl_global_init(3)*, after you are done using libcurl.

This function is thread-safe since libcurl 7.84.0 if *curl_version_info(3)* has the CURL_VERSION_THREADSAFE feature bit set (most platforms).

If this is not thread-safe, you must not call this function when any other thread in the program (i.e. a thread sharing the same memory) is running. This does not just mean no other thread that is using libcurl. Because *curl_global_cleanup(3)* calls functions of other libraries that are similarly thread unsafe, it could conflict with any other thread that uses these other libraries.

See the description in *libcurl(3)* of global environment requirements for details of how to use this function.

CAUTION

curl_global_cleanup(3) does not block waiting for any libcurl-created threads to terminate (such as threads used for name resolving). If a module containing libcurl is dynamically unloaded while libcurl-created threads are still running then your program may crash or other corruption may occur. We recommend you do not run libcurl from any module that may be unloaded dynamically. This behavior may be addressed in the future.

libcurl may not be able to fully clean up after multi-threaded OpenSSL depending on how OpenSSL was built and loaded as a library. It is possible in some rare circumstances a memory leak could occur unless you implement your own OpenSSL thread cleanup. Refer to *libcurl-thread(3)*.

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
int main(void)
{
  curl_global_init(CURL_GLOBAL_DEFAULT);

  /* use libcurl, then before exiting... */

  curl_global_cleanup();
}
```

AVAILABILITY

Added in curl 7.8

RETURN VALUE

None

SEE ALSO

curl_global_init(3), **libcurl**(3), **libcurl-thread**(3)