NAME

curl_global_init - Global libcurl initialization

SYNOPSIS

```
#include <curl/curl.h>

CURLcode curl_global_init(long flags);
```

DESCRIPTION

This function sets up the program environment that libcurl needs. Think of it as an extension of the library loader.

This function must be called at least once within a program (a program is all the code that shares a memory space) before the program calls any other function in libcurl. The environment it sets up is constant for the life of the program and is the same for every program, so multiple calls have the same effect as one call.

The flags option is a bit pattern that tells libcurl exactly what features to init, as described below. Set the desired bits by ORing the values together. In normal operation, you must specify CURL_GLOBAL_ALL. Do not use any other value unless you are familiar with it and mean to control internal operations of 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_init(3)* calls functions of other libraries that are similarly thread unsafe, it could conflict with any other thread that uses these other libraries.

If you are initializing libcurl from a Windows DLL you should not initialize it from *DllMain* or a static initializer because Windows holds the loader lock during that time and it could cause a deadlock.

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

FLAGS

CURL_GLOBAL_ALL

Initialize everything possible. This sets all known bits except **CURL_GLOBAL_ACK_EINTR**.

CURL_GLOBAL_SSL

(This flag's presence or absence serves no meaning since 7.57.0. The description below is for older libcurl versions.)

Initialize SSL.

The implication here is that if this bit is not set, the initialization of the SSL layer needs to be done by the application or at least outside of libcurl. The exact procedure how to do SSL initialization depends on the TLS backend libcurl uses.

Doing TLS based transfers without having the TLS layer initialized may lead to unexpected behaviors.

CURL_GLOBAL_WIN32

Initialize the Win32 socket libraries.

The implication here is that if this bit is not set, the initialization of winsock has to be done by the application or you risk getting undefined behaviors. This option exists for when the initialization is handled outside of libcurl so there is no need for libcurl to do it again.

CURL_GLOBAL_NOTHING

Initialize nothing extra. This sets no bit.

CURL_GLOBAL_DEFAULT

A sensible default. It initializes both SSL and Win32. Right now, this equals the functionality of the **CURL_GLOBAL_ALL** mask.

CURL_GLOBAL_ACK_EINTR

This bit has no point since 7.69.0 but its behavior is instead the default.

Before 7.69.0: when this flag is set, curl acknowledges EINTR condition when connecting or when waiting for data. Otherwise, curl waits until full timeout elapses. (Added in 7.30.0)

EXAMPLE

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

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

```
  curl_global_cleanup();
}
```

AVAILABILITY

Added in 7.8

RETURN VALUE

If this function returns non-zero, something went wrong and you cannot use the other curl functions.

SEE ALSO

curl_easy_init(3), **curl_global_cleanup**(3), **curl_global_init_mem**(3), **curl_global_sslset**(3), **curl_global_trace**(3), **libcurl**(3)