NAME

curl_global_init_mem − global libcurl initialization with memory callbacks

SYNOPSIS

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

CURLcode curl_global_init_mem(long flags,
                              curl_malloc_callback m,
                              curl_free_callback f,
                              curl_realloc_callback r,
                              curl_strdup_callback s,
                              curl_calloc_callback c);
```

DESCRIPTION

This function works exactly as *curl_global_init(3)* with one addition: it allows the application to set callbacks to replace the otherwise used internal memory functions.

If you are using libcurl from multiple threads or libcurl was built with the threaded resolver option then the callback functions must be thread−safe. The threaded resolver is a common build option to enable (and in some cases the default) so we strongly urge you to make your callback functions thread−safe.

All callback arguments must be set to valid function pointers. The prototypes for the given callbacks must match these:

`void *malloc_callback(size_t size);`
: To replace malloc()

`void free_callback(void *ptr);`
: To replace free()

`void *realloc_callback(void *ptr, size_t size);`
: To replace realloc()

`char *strdup_callback(const char *str);`
: To replace strdup()

`void *calloc_callback(size_t nmemb, size_t size);`
: To replace calloc()

This function is otherwise the same as *curl_global_init(3)*, please refer to that man page for documentation.

CAUTION

Manipulating these gives considerable powers to the application to severely screw things up for libcurl. Take care.

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
extern void *malloc_cb(size_t);
extern void free_cb(void *);
extern void *realloc_cb(void *, size_t);
extern char *strdup_cb(const char *);
extern void *calloc_cb(size_t, size_t);

int main(void)
{
  CURLcode result;
```

```
 result = curl_global_init_mem(CURL_GLOBAL_DEFAULT, malloc_cb,
                               free_cb, realloc_cb,
                               strdup_cb, calloc_cb);

 if(result == CURLE_OK) {

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

   curl_global_cleanup();
 }
}
```

AVAILABILITY

Added in curl 7.12.0

RETURN VALUE

This function returns a CURLcode indicating success or error.

CURLE_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*. If *CURLOPT_ERRORBUFFER(3)* was set with *curl_easy_setopt(3)* there can be an error message stored in the error buffer when non−zero is returned.

SEE ALSO

curl_global_cleanup(3), **curl_global_init**(3)