NAME

CURLOPT_PROGRESSFUNCTION − progress meter callback

SYNOPSIS

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

int progress_callback(void *clientp,
                      double dltotal,
                      double dlnow,
                      double ultotal,
                      double ulnow);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_PROGRESSFUNCTION,
                          progress_callback);
```

DESCRIPTION

Pass a pointer to your callback function, which should match the prototype shown above.

This option is deprecated and we encourage users to use the newer *CURLOPT_XFERINFOFUNCTION(3)* instead, if you can.

This function gets called by libcurl instead of its internal equivalent with a frequent interval. While data is being transferred it is invoked frequently, and during slow periods like when nothing is being transferred it can slow down to about one call per second.

clientp is the pointer set with *CURLOPT_PROGRESSDATA(3)*, it is not used by libcurl but is only passed along from the application to the callback.

The callback gets told how much data libcurl is about to transfer and has transferred, in number of bytes. *dltotal* is the total number of bytes libcurl expects to download in this transfer. *dlnow* is the number of bytes downloaded so far. *ultotal* is the total number of bytes libcurl expects to upload in this transfer. *ulnow* is the number of bytes uploaded so far.

Unknown/unused argument values passed to the callback are be set to zero (like if you only download data, the upload size remains 0). Many times the callback is called one or more times first, before it knows the data sizes so a program must be made to handle that.

Return zero from the callback if everything is fine.

If your callback function returns CURL_PROGRESSFUNC_CONTINUE it causes libcurl to continue executing the default progress function.

Return 1 from this callback to make libcurl abort the transfer and return *CURLE_ABORTED_BY_CALLBACK*.

If you transfer data with the multi interface, this function is not called during periods of idleness unless you call the appropriate libcurl function that performs transfers.

CURLOPT_NOPROGRESS(3) must be set to 0 to make this function actually get called.

DEFAULT

NULL. libcurl has an internal progress meter. That is rarely wanted by users.

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
struct progress {
  char *private;
  size_t size;
};

static int progress_callback(void *clientp,
                             double dltotal,
                             double dlnow,
                             double ultotal,
                             double ulnow)
{
  struct progress *memory = clientp;
  printf("private: %p\n", memory->private);

  /* use the values */

  return 0; /* all is good */
}

int main(void)
{
  struct progress data;

  CURL *curl = curl_easy_init();
  if(curl) {
    CURLcode result;
    /* pass struct to callback */
    curl_easy_setopt(curl, CURLOPT_PROGRESSDATA, &data);
    curl_easy_setopt(curl, CURLOPT_PROGRESSFUNCTION, progress_callback);

    result = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
  }
}
```

DEPRECATED

Deprecated since 7.32.0.

AVAILABILITY

Added in curl 7.1

RETURN VALUE

curl_easy_setopt(3) returns a CURLcode indicating success or error.

CURLE_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*.

SEE ALSO

CURLOPT_NOPROGRESS(3), **CURLOPT_VERBOSE**(3), **CURLOPT_XFERINFOFUNCTION**(3)