NAME

CURLOPT_WRITEFUNCTION − callback for writing received data

SYNOPSIS

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

size_t write_callback(char *ptr, size_t size, size_t nmemb, void *userdata);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_WRITEFUNCTION, write_callback);
```

DESCRIPTION

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

This callback function gets called by libcurl as soon as there is data received that needs to be saved. For most transfers, this callback gets called many times and each invoke delivers another chunk of data. *ptr* points to the delivered data, and the size of that data is *nmemb*; *size* is always 1.

The data passed to this function is not null−terminated.

The callback function is passed as much data as possible in all invokes, but you must not make any assumptions. It may be one byte, it may be thousands. The maximum amount of body data that is passed to the write callback is defined in the curl.h header file: *CURL_MAX_WRITE_SIZE* (the usual default is 16K). If *CURLOPT_HEADER(3)* is enabled, which makes header data get passed to the write callback, you can get up to *CURL_MAX_HTTP_HEADER* bytes of header data passed into it. This usually means 100K.

This function may be called with zero bytes data if the transferred file is empty.

Set the *userdata* argument with the *CURLOPT_WRITEDATA(3)* option.

Your callback should return the number of bytes actually taken care of. If that amount differs from the amount passed to your callback function, it signals an error condition to the library. This causes the transfer to get aborted and the libcurl function used returns *CURLE_WRITE_ERROR*.

You can also abort the transfer by returning CURL_WRITEFUNC_ERROR (added in 7.87.0), which makes *CURLE_WRITE_ERROR* get returned.

If the callback function returns CURL_WRITEFUNC_PAUSE it pauses this transfer. See *curl_easy_pause(3)* for further details.

Set this option to NULL to get the internal default function used instead of your callback. The internal default function writes the data to the FILE * given with *CURLOPT_WRITEDATA(3)*.

This option does not enable HSTS, you need to use *CURLOPT_HSTS_CTRL(3)* to do that.

DEFAULT

fwrite(3)

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
#include <stdlib.h> /* for realloc */
#include <string.h> /* for memcpy */

struct memory {
  char *response;
  size_t size;
```

```
};

static size_t cb(char *data, size_t size, size_t nmemb, void *clientp)
{
 size_t realsize = nmemb;
 struct memory *mem = (struct memory *)clientp;

 char *ptr = realloc(mem->response, mem->size + realsize + 1);
 if(!ptr)
   return 0;  /* out of memory */

 mem->response = ptr;
 memcpy(&(mem->response[mem->size]), data, realsize);
 mem->size += realsize;
 mem->response[mem->size] = 0;

 return realsize;
}

int main(void)
{
 struct memory chunk = { 0 };
 CURLcode result;
 CURL *curl = curl_easy_init();
 if(curl) {
   /* send all data to this function */
   curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, cb);

   /* we pass our 'chunk' struct to the callback function */
   curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&chunk);

   /* send a request */
   result = curl_easy_perform(curl);

   /* remember to free the buffer */
   free(chunk.response);

   curl_easy_cleanup(curl);
  }
}
```

HISTORY

Support for the CURL_WRITEFUNC_PAUSE return code was added in version 7.18.0.

AVAILABILITY

Added in curl 7.1

RETURN VALUE

This returns CURLE_OK.

SEE ALSO

CURLOPT_HEADERFUNCTION(3), **CURLOPT_READFUNCTION**(3), **CURLOPT_WRITEDATA**(3)