NAME

CURLOPT_CHUNK_BGN_FUNCTION − callback before a transfer with FTP wildcard match

SYNOPSIS

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

struct curl_fileinfo {
  char *filename;
  curlfiletype filetype;
  time_t time;   /* always zero */
  unsigned int perm;
  int uid;
  int gid;
  curl_off_t size;
  long int hardlinks;

  struct {
    /* If some of these fields is not NULL, it is a pointer to b_data. */
    char *time;
    char *perm;
    char *user;
    char *group;
    char *target; /* pointer to the target filename of a symlink */
  } strings;

  unsigned int flags;

  /* used internally */
  char *b_data;
  size_t b_size;
  size_t b_used;
};

long chunk_bgn_callback(const void *transfer_info, void *ptr,
                        int remains);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_CHUNK_BGN_FUNCTION,
                          chunk_bgn_callback);
```

DESCRIPTION

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

This callback function gets called by libcurl before a part of the stream is going to be transferred (if the transfer supports chunks).

The *transfer_info* pointer points to a **curl_fileinfo** struct with details about the file that is about to get transferred.

This callback makes sense only when using the *CURLOPT_WILDCARDMATCH(3)* option for now.

The target of transfer_info parameter is a "feature depended" structure. For the FTP wildcard download, the target is **curl_fileinfo** structure (see *curl/curl.h*). The parameter *ptr* is a pointer given by *CURLOPT_CHUNK_DATA(3)*. The parameter remains contains number of chunks remaining per the transfer. If the feature is not available, the parameter has zero value.

Return *CURL_CHUNK_BGN_FUNC_OK* if everything is fine, *CURL_CHUNK_BGN_FUNC_SKIP* if you

want to skip the concrete chunk or *CURL_CHUNK_BGN_FUNC_FAIL* to tell libcurl to stop if some error occurred.

DEFAULT

NULL

PROTOCOLS

This functionality affects ftp only

EXAMPLE

```
#include <stdio.h>

struct callback_data {
  FILE *output;
};

static long file_is_coming(struct curl_fileinfo *finfo,
                           void *ptr,
                           int remains)
{
  struct callback_data *data = ptr;
  printf("%3d %40s %10luB ", remains, finfo->filename,
         (unsigned long)finfo->size);

  switch(finfo->filetype) {
  case CURLFILETYPE_DIRECTORY:
    printf(" DIR\n");
    break;
  case CURLFILETYPE_FILE:
    printf("FILE ");
    break;
  default:
    printf("OTHER\n");
    break;
  }

  if(finfo->filetype == CURLFILETYPE_FILE) {
    /* do not transfer files >= 50B */
    if(finfo->size > 50) {
      printf("SKIPPED\n");
      return CURL_CHUNK_BGN_FUNC_SKIP;
    }

    data->output = fopen(finfo->filename, "wb");
    if(!data->output) {
      return CURL_CHUNK_BGN_FUNC_FAIL;
    }
  }

  return CURL_CHUNK_BGN_FUNC_OK;
}

int main()
{
  /* data for callback */
  struct callback_data callback_info;
```

```
 CURL *curl = curl_easy_init();

 /* callback is called before download of concrete file started */
 curl_easy_setopt(curl, CURLOPT_CHUNK_BGN_FUNCTION, file_is_coming);
 curl_easy_setopt(curl, CURLOPT_CHUNK_DATA, &callback_info);
}
```

AVAILABILITY

Added in curl 7.21.0

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_CHUNK_END_FUNCTION(3), **CURLOPT_WILDCARDMATCH**(3)