NAME

curl_formadd - add a section to a multipart/formdata HTTP POST

SYNOPSIS

#include <curl/curl.h>

CURLFORMcode curl_formadd(struct HttpPost ** *firstitem,* **struct HttpPost **** *lastitem,* **...);**

DESCRIPTION

curl_formadd() is used to append sections when building a multipart/formdata HTTP POST (sometimes refered to as rfc1867-style posts). Append one section at a time until you've added all the sections you want included and then you pass the *firstitem* pointer as parameter to **CURLOPT_HTTPPOST**. *lastitem* is set after each call and on repeated invokes it should be left as set to allow repeated invokes to find the end of the list faster.

After the *lastitem* pointer follow the real arguments.

The pointers **firstitem* and **lastitem* should both be pointing to NULL in the first call to this function. All list-data will be allocated by the function itself. You must call *curl_formfree* after the form post has been done to free the resources again.

First, there are some basics you need to understand about multipart/formdata posts. Each part consists of at least a NAME and a CONTENTS part. If the part is made for file upload, there are also a stored CONTENT-TYPE and a FILENAME. Below here, we'll discuss on what options you use to set these properties in the parts you want to add to your post.

OPTIONS

CURLFORM_COPYNAME followed by string is used to set the name of this part. libcurl copies the given data, so your application doesn't need to keep it around after this function call. If the name isn't zero terminated properly, or if you'd like it to contain zero bytes, you need to set the length of the name with **CURLFORM_NAMELENGTH**.

CURLFORM_PTRNAME followed by a string is used for the name of this part. libcurl will use the pointer and refer to the data in your application, you must make sure it remains until curl no longer needs it. If the name isn't zero terminated properly, or if you'd like it to contain zero bytes, you need to set the length of the name with **CURLFORM_NAMELENGTH**.

CURLFORM_COPYCONTENTS followed by a string is used for the contents of this part, the actual data to send away. libcurl copies the given data, so your application doesn't need to keep it around after this function call. If the data isn't zero terminated properly, or if you'd like it to contain zero bytes, you need to set the length of the name with **CURLFORM_CONTENTSLENGTH**.

CURLFORM_PTRCONTENTS followed by a string is used for the contents of this part, the actual data to send away. libcurl will use the pointer and refer to the data in your application, you must make sure it remains until curl no longer needs it. If the data isn't zero terminated properly, or if you'd like it to contain zero bytes, you need to set the length of the name with **CURLFORM_CONTENTSLENGTH**.

CURLFORM_FILECONTENT followed by a file name, makes that file read and the contents will be used in as data in this part.

CURLFORM_FILE followed by a file name, makes this part a file upload part. It sets the file name field to the actual file name used here, it gets the contents of the file and passes as data and sets the content-type if the given file match one of the new internally known file extension. For **CURLFORM_FILE** the user may send one or more files in one part by providing multiple **CURLFORM_FILE** arguments each followed by the filename (and each CURLFORM_FILE is allowed to have a CURLFORM_CONTENT-TYPE).

CURLFORM_CONTENTTYPE followed by a pointer to a string with a content-type will make curl use this given content-type for this file upload part, possibly instead of an internally chosen one.

CURLFORM_FILENAME followed by a pointer to a string to a name, will make libcurl use the given name in the file upload part, intead of the actual file name given to *CURLFORM_FILE*.

BCURLFORM_BUFFER followed by a string, tells libcurl that a buffer is to be used to upload data instead of using a file. The given string is used as the value of the file name field in the content header.

CURLFORM_BUFFERPTR followed by a pointer to a data area, tells libcurl the address of the buffer containing data to upload (as indicated with *CURLFORM_BUFFER*). The buffer containing this data must not be freed until after curl_easy_cleanup is called.

CURLFORM_BUFFERLENGTH followed by a long with the size of the *CURLFORM_BUFFERPTR* data area, tells libcurl the length of the buffer to upload.

CURLFORM_ARRAY Another possibility to send options to curl_formadd() is the **CURLFORM_ARRAY** option, that passes a struct curl_forms array pointer as its value. Each curl_forms structure element has a CURLformoption and a char pointer. The final element in the array must be a CURLFORM_END. All available options can be used in an array, except the CURLFORM_ARRAY option itself! The last argument in such an array must always be **CURLFORM_END**.

CURLFORM_CONTENTHEADER specifies extra headers for the form POST section. This takes a curl_slist prepared in the usual way using **curl_slist_append** and appends the list of headers to those libcurl automatically generates. The list must exist while the POST occurs, if you free it before the post completes you may experience problems.

When you've passed the HttpPost pointer to *curl_easy_setopt* (using the *CURLOPT_HTTPPOST* option), you must not free the list until after you've called *curl_easy_cleanup* for the curl handle.

See example below.

RETURN VALUE

0 means everything was ok, non-zero means an error occurred as *<curl/curl.h>* defines.

EXAMPLE

```
struct HttpPost* post = NULL;
struct HttpPost* last = NULL;
char namebuffer[] = "name buffer";
long namelength = strlen(namebuffer);
char buffer[] = "test buffer";
char htmlbuffer[] = "<HTML>test buffer</HTML>";
long htmlbufferlength = strlen(htmlbuffer);
struct curl_forms forms[3];
char file1[] = "my-face.jpg";
char file2[] = "your-face.jpg";
/* add null character into htmlbuffer, to demonstrate that
   transfers of buffers containing null characters actually work
*/
htmlbuffer[8] = '\0';

/* Add simple name/content section */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "name",
             CURLFORM_COPYCONTENTS, "content", CURLFORM_END);

/* Add simple name/content/contenttype section */
```

```
curl_formadd(&post, &last, CURLFORM_COPYNAME, "htmlcode",
        CURLFORM_COPYCONTENTS, "<HTML></HTML>",
        CURLFORM_CONTENTTYPE, "text/html", CURLFORM_END);

/* Add name/ptrcontent section */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "name_for_ptrcontent",
        CURLFORM_PTRCONTENTS, buffer, CURLFORM_END);

/* Add ptrname/ptrcontent section */
curl_formadd(&post, &last, CURLFORM_PTRNAME, namebuffer,
            CURLFORM_PTRCONTENTS, buffer, CURLFORM_NAMELENGTH,
            namelength, CURLFORM_END);

/* Add name/ptrcontent/contenttype section */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "html_code_with_hole",
        CURLFORM_PTRCONTENTS, htmlbuffer,
        CURLFORM_CONTENTSLENGTH, htmlbufferlength,
        CURLFORM_CONTENTTYPE, "text/html", CURLFORM_END);

/* Add simple file section */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "picture",
        CURLFORM_FILE, "my-face.jpg", CURLFORM_END);

/* Add file/contenttype section */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "picture",
        CURLFORM_FILE, "my-face.jpg",
        CURLFORM_CONTENTTYPE, "image/jpeg", CURLFORM_END);

/* Add two file section */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "pictures",
        CURLFORM_FILE, "my-face.jpg",
        CURLFORM_FILE, "your-face.jpg", CURLFORM_END);

/* Add two file section using CURLFORM_ARRAY */
forms[0].option = CURLFORM_FILE;
forms[0].value  = file1;
forms[1].option = CURLFORM_FILE;
forms[1].value  = file2;
forms[2].option  = CURLFORM_END;

/* Add a buffer to upload */
curl_formadd(&post, &last,
        CURLFORM_COPYNAME, "name",
        CURLFORM_BUFFER, "data",
        CURLFORM_BUFFERPTR, record,
        CURLFORM_BUFFERLENGTH, record_length,
        CURLFORM_END);

/* no option needed for the end marker */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "pictures",
        CURLFORM_ARRAY, forms, CURLFORM_END);
/* Add the content of a file as a normal post text value */
curl_formadd(&post, &last, CURLFORM_COPYNAME, "filecontent",
        CURLFORM_FILECONTENT, ".bashrc", CURLFORM_END);
```

```
 /* Set the form info */
 curl_easy_setopt(curl, CURLOPT_HTTPPOST, post);
```

SEE ALSO

curl_easy_setopt(3), **curl_formparse**(3) [deprecated], **curl_formfree**(3)

BUGS

Surely there are some, you tell me!