NAME

libcurl-tutorial − libcurl programming tutorial

Objective

This document attempts to describe the general principles and some basic approaches to consider when programming with libcurl. The text will focus mainly on the C interf

Portable Code in a Portable World

The people behind libcurl have put a considerable effort to make l

first, so please continue reading for better understanding.

To u

(*CURLOPT_WRITEDATA* was formerly known as *CURLOPT_FILE*. Both names still work and do the same thing).

PolarSSL

Required actions unknown.

yassl

Required actions unknown.

axTLS

Required actions unknown.

```
size_t function(char *bufptr, size_t size, size_t nitems, void *userp);
```

Where bufptr is the pointer to a buffer we fill in with data to upload and size*nitems is the size of the buffer and therefore also the maximum amount of data we can return to libcurl in this call. The 'userp' pointer is the custom pointer we set to point to a struct of ours to pass private data between the application and the callback.

```
curl_easy_setopt(easyhandle, CURLOPT_READFUNCTION, read_function);

curl_easy_setopt(easyhandle, CURLOPT_READDATA, &filedata);
```

Tell libcurl that we want to upload:

```
curl_easy_setopt(easyhandle, CURLOPT_UPLOAD, 1L);
```

A few protocols won't behave properly when uploads are done without any prior knowledge of the expected file size. So, set the upload file size using the CURLOPT_INFILESIZE_LARGE for all known file sizes like this[1]:

```
/* in this example, file_size must be an curl_off_t variable */
curl_easy_setopt(easyhandle, CURLOPT_INFILESIZE_LARGE, file_size);
```

When you call *curl_easy_perform(3)* this time, it'll perform all the necessary operations and when it has invoked the upload it'll call your supplied callback to get the data to upload. The program should return as much data as possible in every invoke, as tRLOPTj /R15 91.2199Rwc74r -238.36 -24 Tm.lAe necesss fa_ofn e

```
        CURLFORM_COPYNAME, "logotype-image",
        CURLFORM_FILECONTENT, "curl.png", CURLFORM_END);

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

curl_easy_perform(easyhandle); /* post away! */

/* free the post data again */
curl_formfree(post);
```

Multipart formposts are chains of parts using MIME-style separators and headers. It means that each one of these separate parts get a few headers set that describe the individual content-type, size etc. T

This is howeve

Other interesting details that improve p

```
struct curl_slist *headers=NULL; /* init to NULL is important */

headers = curl_slist_append(headers, "Hey-server-hey: how are you?");
headers = curl_slist_append(headers, "X-silly-content: yes");

/* pass our list of custom made headers */
curl_easy_setopt(easyhandle, CURLOPT_HTTPHEADER, headers);

curl_easy_perform(easyhandle); /* transfer http */

curl_slist_free_all(headers); /* free the header list */
```

Authentication

Use of CURLOPT_UNRESTRICTED_AUTH could cause authentication information to be sent to an unknown second server. Apps can mitigate against this by disabling CURLOPT_FOLLOWLOCATION and handling redirects itself, sanitizing where necessary.

Use of the CURLAUTH_ANY option to CURLOPT_HTTPAUTH could result in user name and password being sent in clear text to an HTTP server. Instead, use CURLAUTH_ANYSAFE which ensures that the password is encrypted o

generate a file name. An application could also use CURLINFO_EFFECTIVE_URL to generate a file name from a server

some time-out code so we advise you to never u