NAME

libcurl-tutorial − libcurl programming tutorial

Objective

This document attempts to describe the general principles and some basic approaches to consider when p
gramming with libcurl. The text will focus mainly on the C interf might apply fairly well on other
interfaces as well as usually follow the C one pretty closely.

This document will refer to 'the user' as the person writing the source code that uses libcurl. That w
probably be you or someone in your position. What will be generally referred to as '

standard replies that confuse the library which then confuses your program.

There's one golden rule when these things occur: set the CURLOPT_VERBOSE option to TRUE. It'll cause the library to spew out the entire protocol details it sends, some internal info and some received protocol data as well (especially when using FTP). If you're using HTTP

```
/* post binary data */
curl_easy_setopt(easyhandle, CURLOPT_POSTFIELDS, binaryptr);

/* set the size of the postfields data */
curl_easy_setopt(easyhandle, CURLOPT_POSTFIELDSIZE, 23);

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

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

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

While the simple examples above cover t

```
curl_formadd(&post, &last,
        CURLFORM_COPYNAME, "logotype-image",
        CURLFORM_FILECONTENT, "curl.xml",
```

a proxy even though a variable may say so. If 'no_proxy' is a plain asterisk ("*") it matches all hosts.

SSL and Proxies

SSL is for secure point-to-point connections. This involv

and execute that.

- Realialc ja vascript code and rewrite the same logic in another language.

- Implement a javascript interpreted, people have s

HTTP Headers Used by libcurl

When you use libcurl to do HTTP requests, it'll pass along a series of headers automatically. It might be good for you to know and understand these ones.

```
curl_easy_perform(easyhandle); /* transfer ftp data! */

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

enough, and you may not have to save the cookies to disk at all. Note that the file you specify to CURLOPT_COOKIEFILE doesn't have to exist to enable the parser, so a common way to just enable the parser and not read able might be to use a file name you know doesn't exist.

If you rather use existing cookies that you've previously received with your Netscape or Mozilla browsers, you can make libcurl use that cookie file as input. The CURLOPT_COOKIEFILE is used for that too, as libcurl will automatically find out what kind of file it is and act accordingly.

The perhaps most advanced cookie operatiozin3.296 Tcf

"Headers" for FTP transfers equal all the FTP server responses. They aren't actually true headers, but in this case we pretend they are! ;-)

Post Transfer Information

[curl_easy_getinfo]

Security Considerations

libcurl is in itself not insecure. If used the right way, you can use libcurl to transfer data pretty safely.

There are of course many things to consider that may loosen up this situation:

Command Lines

the same time, without forcing you to use multiple threads.

To use this interface, you are better off if you first understand the basics of how to use the easy interface. The multi interface is simply a way to make multiple transfers at the same time, by adding up multiple easy handles in to a "multi stack".

You create the easy handles you want and you set all the options just like you have been told above, and then you create a multi handle with curl_multi_init(3) and add all those easy handles to that multi handle with curl_multi_add_handle(3).

When you've added the handles you have for the moment (you can still add new ones at any time), you start the transfers by call curl_multi_perform(3).

curl_multi_perform(3) is asynchronous. It will only exe