<https://blog.csdn.net/u010487568/article/details/52958367>

## 简介

libcurl作为是一个多协议的便于客户端使用的URL传输库，基于C语言，提供C语言的API接口，支持DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet and TFTP这些协议，同时支持使用SSL证书的安全文件传输：HTTP POST, HTTP PUT, FTP 上传, 基于HTTP形式的上传、代理、Cookies、用户加密码的认证等多种应用场景。另外，libcurl是一个高移植性的库，能在绝大多数系统上运行，包括Solaris, NetBSD, FreeBSD, OpenBSD, Darwin, HPUX, IRIX, AIX, Tru64, Linux, UnixWare, HURD, Windows, Amiga, OS/2, BeOs, Mac OS X, Ultrix, QNX, OpenVMS, RISC OS, Novell NetWare, DOS等。

前段时间需要使用C++发起http请求，这里对这些使用做一个总结以备忘。libcurl封装了协议的底层实现，非常方便用户开发应用层逻辑。

## 使用步骤

具体使用主要遵循如下步骤：   
1. 全局初始化   
2. 创建当次请求句柄   
3. 设置属性   
4. 执行   
5. 销毁当次请求句柄   
6. 全局析构

下面详细说明：

### 1.全局初始化

CURLcode curl\_global\_init(long flags);

flags:

CURL\_GLOBAL\_ALL //初始化所有内部的模块

CURL\_GLOBAL\_SSL //初始化支持安全套接字

CURL\_GLOBAL\_WIN32 //初始化win32套接字库

CURL\_GLOBAL\_NOTHING //不初始化任何额外模块

CURL\_GLOBAL\_DEFAULT //与CURL\_GLOBAL\_ALL相同

该函数必须再所有其他libcurl函数调用前被调用，从而构建整个libcurl函数运行所需的环境，多次调用是幂等的。flags参数能够使用”或”操作符进行多个选项的拼接。一般情况使用CURL\_GLOBAL\_ALL是最好的选择。   
该函数不是线程安全的，不能在程序的其他线程中调用，只能应用程序开始时，进行全局初始化调用。虽然不调用这个函数，使用curl\_easy\_init也会自行调用该函数，但在多线程处理时，可能出现多次调用的情况，应避免。

### 2.创建当次请求句柄

每次请求都需要创建一个句柄，所有操作围绕以此句柄进行：

CURL \*curl\_handler = curl\_easy\_init();

### 3.设置属性

libcurl针对所有协议，统一使用一个简单的函数curl\_easy\_setopt进行设置，这大大简化了使用，不过需要参考不同协议设置不同的属性<https://curl.haxx.se/libcurl/c/curl_easy_perform.html>。这里以http发送GET请求为例设置如下：

//设置请求的url

curl\_easy\_setopt(curl\_handler, CURLOPT\_URL, url);

//设置是否返回请求头

curl\_easy\_setopt(curl\_handler, CURLOPT\_HEADER, 1L);

//设置屏蔽其他信号

curl\_easy\_setopt(curl\_handler, CURLOPT\_NOSIGNAL, 1L);

//设置下载数据回调函数

curl\_easy\_setopt(curl\_handler, CURLOPT\_WRITEFUNCTION, write\_func);

curl\_easy\_setopt(curl\_handler, CURLOPT\_WRITEDATA, write\_data);

回调函数原型：

size\_t function( void \*ptr, size\_t size, size\_t nmemb, void \*userp);

函数将在libcurl接收到数据后被调用。

void \*ptr是下载回来的数据.

void \*userp是用户指针, 用户通过这个指针传输自己的数据.

CURLOPT\_WRITEDATA：设置回调函数中的void \*userp指针的来源。

//设置是否使用下载进度控制函数

curl\_easy\_setopt(curl\_handler, CURLOPT\_NOPROGRESS, 0L);

curl\_easy\_setopt(curl\_handler, CURLOPT\_PROGRESSFUNCTION, prog\_func);

curl\_easy\_setopt(curl\_handler, CURLOPT\_PROGRESSDATA, pdata);

下载进度回调函数与下载数据的回调函数原型相同，data也相同。

//设置请求头

struct curl\_list \*header\_list = NULL;

for (int i = 0; i < headers.size(); ++i) {

header\_list = curl\_slist\_append(header\_list, headers[i].c\_str());

}

curl\_easy\_setopt(curl\_handler, CURLOPT\_HTTPHEADER, header\_list);

curl\_slist\_free\_all(header\_list);

//其他选项

CURLOPT\_HEADERFUNCTION

CURLOPT\_HEADERDATA

只取HTTP头部数据, 处理与下载数据回调的处理相同.

CURLOPT\_TIMEOUT

超时时间.

CURLOPT\_CONNECTIONTIMEOUT

连接等待时间.

CURLOPT\_FOLLOWLOCATION

　　 设置支持302重定向

　　 CURLOPT\_RANGE

　 断点续传, 指定传输分片, 格式:"0-200"

其中，针对http协议，不同请求有特定的属性需要设置，下面依次列举：   
- HEAD：需要设置CURLOPT\_NOBODY为true   
- DELETE：需要设置CURLOPT\_CUSTOMREQUEST为”DELETE”   
- PUT：需要设置CURLOPT\_UPLOAD为true，同时设置CURLOPT\_READFUNCTION、CURLOPT\_READDATA、CURLOPT\_INFILESIZE\_LARGE   
- POST：需要设置CURLOPT\_POST为true，设置CURLOPT\_POSTFIELDS、CURLOPT\_POSTFIELDSIZE

### 4.执行

CURLcode curl\_easy\_perform(CURL \*handler);

该函数执行当次请求创建的句柄，返回值有非常详细的定义[libcurl库返回状态码解释与速查](http://blog.csdn.net/u010487568/article/details/52624477)。

### 5.销毁当次请求句柄

void curl\_easy\_cleanup(CURL \*handler);

该函数销毁当次请求创建的句柄。

### 6.全局析构

void curl\_global\_cleanup();

该函数销毁全局执行环境。

## 其他接口

libcurl除了上述常用接口外，提供了其他接口可以进行更为方便和精确的控制。

CURLcode curl\_easy\_getinfo(CURL \*curl, CURLINFO info, ... );

//支持的CURLINFO类型见此：https://curl.haxx.se/libcurl/c/curl\_easy\_getinfo.html

curl\_version() returns a pointer to the libcurl version string

curl\_getdate() converts a date string to time\_t

curl\_formadd() build multipart form-data posts

curl\_formfree() free a previously built form POST

curl\_slist\_append() builds a linked list

curl\_slist\_free\_all() frees a whole curl\_slist as made with curl\_slist\_append()

curl\_easy\_escape() URL encodes a string

curl\_easy\_unescape() URL decodes a string

后续将实现的例程进行更新。