<https://blog.csdn.net/wangqing_12345/article/details/52233067>

#include <curl/curl.h>  
CURLcodecurl\_easy\_setopt(CURL \*handle, CURLoption option, parameter);  
说明：  
此函数用来告诉 libcurl 执行什么样的动作。该函数有 3 个参数(该函数的可设置选项非常之多)：  
第 1 个参数 handle 是由 curl\_easy\_init() 返回的句柄；第 2 个参数是可以设置的选项(CURLoption)；第 3 个参数是与第 2 个选项相关联的参数，这个参数可以是 long 型，也可以是一个函数指针(function pointer)，还可以是一个对象的指针 (object pointer)，或者是一个 curl\_off\_t 类型，这些参数类型必须由选项值(CURLoption)来确定。  
  
具体选项说明(分大类和小类，本说明为个人理解以及对说明手册的翻译，根据应用不断更新说明内容，分类顺序不按照说明手册)：  
  
网络选项(NETWORK OPTIONS)：  
  
1. CURLOPT\_URL  
这个选项后面接实际要处理的 URL ，这个 URL 是一个以 '\0' 结尾的字符串或参数指针(关于这个参数的简单示例见：<http://www.groad.net/bbs/read.php?tid-1641.html>)。如果 URL 参数不写上协议头(如 "http://" 或者 "ftp:// 等等 )，那么函数会自己进行猜解所给的主机上用的是哪一种服务协议。假如你给的这个地址是一个不被 支持的协议，那么在其后执行 [curl\_easy\_perform()](http://www.groad.net/bbs/read.php?tid-1644.html) 函数或 curl\_multi\_perform() 函数时， libcurl 将返回错误(CURLE\_UNSUPPORTED\_PROTOCOL)。这个选项是唯一一个必须在 [curl\_easy\_perform()](http://www.groad.net/bbs/read.php?tid-1644.html) 调用之前就要设置的选项。  
  
2. CURLOPT\_POSTFIELDSIZE  
该选项要求第 3 个参数 parameter 是一个 void \* 指针，它指向一个向 HTTP 服务器 POST 出去的一段数据，这段数据要根据服务器的具体要求填写。该选项用法参考示例：<http://www.groad.net/bbs/read.php?tid-3909.html>  
  
3. CURLOPT\_WRITEFUNCTION  
使用该选项时，要求第 3 个参数中的回调函数必须是下面的函数原型：

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

在启动会话时，一旦检测到有需要接收的数据时，回调函数被调用。ptr 所指向的数据大小由 size 和 nmemb 的乘积获得。函数最后需要返回接收数据的大小。如果不使用该函数，那么接收到的数据会直接打印到终端；使用该函数，那么接收到的数据保存在 ptr 所执向的区域，可以利用此来保存接收下来的数据。  
  
4. CURLOPT\_USERAGENT  
该选项要求传递一个以 '\0' 结尾的字符串指针，这个字符串用来在向服务器请求时发送 HTTP 头部中的 User-Agent 信息，有些服务器是需要检测这个信息的，如果没有设置 User-Agent，那么服务器拒绝请求。设置后，可以骗过服务器对此的检查。  
  
5.  CURLOPT\_WRITEDATA  
  
使用该选项时，第 3 个参数作为用户数据的指针而传递到使用  CURLOPT\_WRITEFUNCTION 选项时指定的回调函数中(第 4 个参数)。如果不想用回调函数而保存数据，那么可以使用 CURLOPT\_WRITEDATA 选项，使用该选项时，函数的第 3 个参数必须是个 FILE 指针，函数会将接收到的数据自动的写到这个 FILE 指针所指向的文件流中。  
  
6. CURLOPT\_VERBOSE  
在使用该选项且第 3 个参数为 1 时，curl 库会显示详细的操作信息。这对程序的调试具有极大的帮助。  
  
7. CURLOPT\_NOBODY  
使用该选项时，若第 3 个参数设为 1，这样在输出中就不会包含主体内容部分。这仅是对在传输的所有内容中，含有“头部”和“主题内容”两部分的协议而言。如 HTTP(S) 服务器，在这种情况下将会使 libcurl 库仅发出一个头部请求。  
  
8. CURLOPT\_HEADER  
使用该选项时，第 3 个参数设置为 1，那么会通知 curl 库在输出时要同时包含 "头部“ 和 "主题内容" 两个部分。该选项仅是对那些同时包含了”头部“和”主题内容“这两部分的协议而言(如 HTTP)。  
  
9. CURLOPT\_HEADERFUNCTION  
该选项与上面第 3 个选项 CURLOPT\_WRITEFUNCTION 类似，只要它一接收到头部信息时，它就会执行回调函数。需要注意的是，回调函数里处理的头部包含了所有收到的响应的头部信息，而不只是最后一次的响应。如果需要处理其中的一个头部，那么自己需要在所收集的头部信息中进行区分。  
  
10. CURLOPT\_WRITEHEADER 和  CURLOPT\_HEADERDATA  
这两个选项是同一种意思。它们和第 5 条中的 CURLOPT\_WRITEDATA 选项功能一样，表示在接收到头部信息并调用回调函数时，给回调函数传递第 4 个参数。  
  
11. CURLOPT\_INFILESIZE  
当向服务器上传文件时，该选项用来告诉 curl 库期望上传的文件的大小。使用该选项时，应该给函数第 3 个参数的应该是个 long 型变量。如果用的是 SCP 传输，那么该选项强制使用 CURLOPT\_INFILESIZE\_LARGE 。  
  
12. CURLOPT\_INFILESIZE\_LARGE  
该选项和 CURLOPT\_INFILESIZE 功能一样，但是它要求函数的第 3 个参数必须是个 curl\_off\_t 类型。curl\_off\_t 为 int64\_t 类型，而 int64\_t 定义在 stdint.h 中：

if \_\_WORDSIZE == 64

typedef long int                int64\_t;

# else

\_\_extension\_\_

typedef long long int           int64\_t;

由上面定义知，一般的，如果是 32 位平台，它就是 long long 型，表示 64 位的；如果是 64 位平台，它就是 long 型，也为 64 位。总之，就是个 64 位的。  
  
13. CURLOPT\_QUOTE 和 CURLOPT\_POSTQUOTE  
这两个选项的功能类似，它们的共同点都是给 FTP 或 SFTP 传递命令。这些命令应该放在 struct slist 链表中存储，使用时需要用 curl\_slist\_append() 函数将这些命令打包起来，然后一起发送出去。  
它们的不同点是：CURLOPT\_QUOTE 选项要求命令要在 FTP 传输请求之前就要发送到库，而 CURLOPT\_POSTQUOTE 则可以在 FTP 传输请求发送完后发送。比如像下面的执行顺序是无法达到目的的：

1 . ... ...

2 curl\_easy\_perform(curl);

3 curl\_easy\_setopt(curl, CURLOPT\_QUOTE, headerlist);

4 curl\_easy\_perform(curl);

5 ... ...

上面假设第 1 条 curl\_easy\_perform(curl); 语句已经发送了 FTP 传输请求，然后再用 CURLOPT\_QUOTE 发送命令是错误的，这里应该使用 CURLOPT\_POSTQUOTE 选项。使用 CURLOPT\_QUOTE 选项需要先用 curl\_easy\_perform() 函数将其发送，然后再用一次 curl\_easy\_perform() 发送 FTP 传输请求。也就是说需要执行两次  curl\_easy\_perform() 函数。而用  CURLOPT\_POSTQUOTE 选项则不需要这样，它只要将选项设置好后，然后只执行一次 curl\_easy\_perform() 函数即可。  
  
14. CURLOPT\_READFUNCTION 和 CURLOPT\_READDATA  
这两个选项和上面的  CURLOPT\_WRITEFUNCTION 和 CURLOPT\_WRITEDATA 类似。在 CURLOPT\_READFUNCTION 的回调函数里，第 1 个参数 ptr 指针用来接收从第 4 个参数传递过来的数据(这个参数往往是个文件流指针)，而这个参数是使用 CURLOPT\_READDATA 选项时传递过来的。  
  
15. CURLOPT\_UPLOAD  
在使用该选项时，第 3 个参数设置为 1，表示要准备上传文件。这个参数往往会配合 CURLOPT\_READDATA，CURLOPT\_INFILESIZE\_LARGE，以及 CURLOPT\_INFILESIZE 这几个选项一起用。如果是使用 HTTP 协议，那么使用 PUT 的方法进行上传，除非另有指定。

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

CURLOPT\_HEADER：设为1，则在返回的内容里包含http header；

CURLOPT\_FOLLOWLOCATION：设为0，则不会自动301，302跳转；

\*CURLOPT\_INFILESIZE: 当你上传一个文件到远程站点，这个选项告诉PHP你上传文件的大小。  
\*CURLOPT\_VERBOSE: 如果你想CURL报告每一件意外的事情，设置这个选项为一个非零值。  
\*CURLOPT\_HEADER: 如果你想把一个头包含在输出中，设置这个选项为一个非零值。  
\*CURLOPT\_NOPROGRESS: 如果你不会PHP为CURL传输显示一个进程条，设置这个选项为一个非零值。

注意：PHP自动设置这个选项为非零值，你应该仅仅为了调试的目的来改变这个选项。

\*CURLOPT\_NOBODY: 如果你不想在输出中包含body部分，设置这个选项为一个非零值。  
\*CURLOPT\_FAILONERROR: 如果你想让PHP在发生错误(HTTP代码返回大于等于300)时，不显示，设置这个选项为一人非零值。默认行为是返回一个正常页，忽略代码。  
\*CURLOPT\_UPLOAD: 如果你想让PHP为上传做准备，设置这个选项为一个非零值。  
\*CURLOPT\_POST: 如果你想PHP去做一个正规的HTTP POST，设置这个选项为一个非零值。这个POST是普通的 application/x-www-from-urlencoded 类型，多数被HTML表单使用。  
\*CURLOPT\_FTPLISTONLY: 设置这个选项为非零值，PHP将列出FTP的目录名列表。  
\*CURLOPT\_FTPAPPEND: 设置这个选项为一个非零值，PHP将应用远程文件代替覆盖它。  
\*CURLOPT\_NETRC: 设置这个选项为一个非零值，PHP将在你的 ~./netrc 文件中查找你要建立连接的远程站点的用户名及密码。  
\*CURLOPT\_FOLLOWLOCATION: 设置这个选项为一个非零值(象 “Location: “)的头，服务器会把它当做HTTP头的一部分发送(注意这是递归的，PHP将发送形如 “Location: “的头)。  
\*CURLOPT\_PUT: 设置这个选项为一个非零值去用HTTP上传一个文件。要上传这个文件必须设置CURLOPT\_INFILE和CURLOPT\_INFILESIZE选项.  
\*CURLOPT\_MUTE: 设置这个选项为一个非零值，PHP对于CURL函数将完全沉默。  
\*CURLOPT\_TIMEOUT: 设置一个长整形数，作为最大延续多少秒。  
\*CURLOPT\_LOW\_SPEED\_LIMIT: 设置一个长整形数，控制传送多少字节。  
\*CURLOPT\_LOW\_SPEED\_TIME: 设置一个长整形数，控制多少秒传送CURLOPT\_LOW\_SPEED\_LIMIT规定的字节数。  
\*CURLOPT\_RESUME\_FROM: 传递一个包含字节偏移地址的长整形参数，(你想转移到的开始表单)。  
\*CURLOPT\_SSLVERSION: 传递一个包含SSL版本的长参数。默认PHP将被它自己努力的确定，在更多的安全中你必须手工设置。  
\*CURLOPT\_TIMECONDITION: 传递一个长参数，指定怎么处理CURLOPT\_TIMEVALUE参数。你可以设置这个参数为TIMECOND\_IFMODSINCE 或 TIMECOND\_ISUNMODSINCE。这仅用于HTTP。  
\*CURLOPT\_TIMEVALUE: 传递一个从1970-1-1开始到现在的秒数。这个时间将被CURLOPT\_TIMEVALUE选项作为指定值使用，或被默认TIMECOND\_IFMODSINCE使用。

下列选项的值将被作为字符串：

\*CURLOPT\_URL: 这是你想用PHP取回的URL地址。你也可以在用curl\_init()函数初始化时设置这个选项。  
\*CURLOPT\_USERPWD: 传递一个形如[username]:[password]风格的字符串,作用PHP去连接。  
\*CURLOPT\_PROXYUSERPWD: 传递一个形如[username]:[password] 格式的字符串去连接HTTP代理。  
\*CURLOPT\_RANGE: 传递一个你想指定的范围。它应该是”X-Y”格式，X或Y是被除外的。HTTP传送同样支持几个间隔，用逗句来分隔(X-Y,N-M)。  
\*CURLOPT\_POSTFIELDS: 传递一个作为HTTP “POST”操作的所有数据的字符串。  
\*CURLOPT\_REFERER: 在HTTP请求中包含一个”referer”头的字符串。  
\*CURLOPT\_USERAGENT: 在HTTP请求中包含一个”user-agent”头的字符串。  
\*CURLOPT\_FTPPORT: 传递一个包含被ftp “POST”指令使用的IP地址。这个POST指令告诉远程服务器去连接我们指定的IP地址。这个字符串可以是一个IP地址，一个主机名，一个网络界面名(在UNIX下)，或是‘-’(使用系统默认IP地址)。  
\*CURLOPT\_COOKIE: 传递一个包含HTTP cookie的头连接。  
\*CURLOPT\_SSLCERT: 传递一个包含PEM格式证书的字符串。  
\*CURLOPT\_SSLCERTPASSWD: 传递一个包含使用CURLOPT\_SSLCERT证书必需的密码。  
\*CURLOPT\_COOKIEFILE: 传递一个包含cookie数据的文件的名字的字符串。这个cookie文件可以是Netscape格式，或是堆存在文件中的HTTP风格的头。  
\*CURLOPT\_CUSTOMREQUEST: 当进行HTTP请求时，传递一个字符被GET或HEAD使用。为进行DELETE或其它操作是有益的，更Pass a string to be used instead of GET or HEAD when doing an HTTP request. This is useful for doing or another, more obscure, HTTP request.

注意: 在确认你的服务器支持命令先不要去这样做。

下列的选项要求一个文件描述(通过使用fopen()函数获得)：

\*CURLOPT\_FILE: 这个文件将是你放置传送的输出文件，默认是STDOUT.  
\*CURLOPT\_INFILE: 这个文件是你传送过来的输入文件。  
\*CURLOPT\_WRITEHEADER: 这个文件写有你输出的头部分。  
\*CURLOPT\_STDERR: 这个文件写有错误而不是stderr。

几种选项测试函数:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <curl/curl.h>

#include <curl/easy.h>

static size\_t save\_header(void \*ptr, size\_t size, size\_t nmemb, void \*data)

{

fwrite(ptr, size, nmemb, data);

return (size \* nmemb);

}

int main(void)

{

char url[] = "http://www.sina.com.cn/";

CURL \*curl;         CURLcode res;

FILE \*fp;

if (!(fp = fopen("htmheader.html", "w")))

{

printf ("fopen error\n");

return -1;

}

curl\_global\_init(CURL\_GLOBAL\_ALL);

curl = curl\_easy\_init();

if (curl)

{

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

curl\_easy\_setopt(curl, CURLOPT\_NOBODY, 1L);

curl\_easy\_setopt(curl, CURLOPT\_HEADER, 0L);

curl\_easy\_setopt(curl, CURLOPT\_HEADERFUNCTION, save\_header);

curl\_easy\_setopt(curl, CURLOPT\_WRITEHEADER, fp);

curl\_easy\_perform(curl);

}

curl\_easy\_cleanup(curl);

curl\_global\_cleanup();

fclose (fp);

return 0;

}