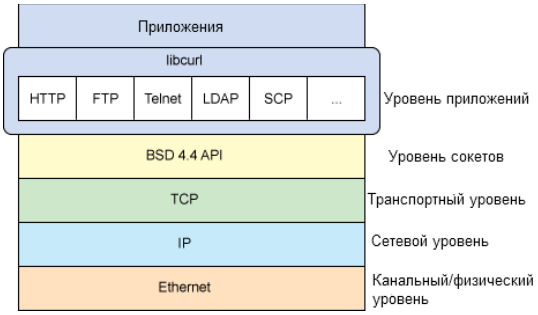
Библиотека libcurl в C++

Запись от Avazart размещена 07.02.2013 в 00:46
Обновил(-а) Avazart 04.12.2013 в 18:44

Цитата:

> **libcurl** это свободная и простая в использовании клиентская библиотека по передачи данных по URL, она поддерживает DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET и TFTP. Также libcurl имеет поддержку SSL сертификатов, HTTP POST, HTTP PUT, FTP загрузку, HTTP form загрузку, proxy, cookies, user+password авторизацию (Basic, Digest, NTLM, Negotiate, Kerberos), докачивания файлов, http прокси туннелирования и многого другого!
>
> libcurl легко переносима, она собирается и работает на многих платформах, включая 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 и остальные...
>
> libcurl свободна, потокобезопасна, совместима с IPv6, функциональна, имеет хорошую поддержку, быстрая, тщательно задокументирована и уже используется во многих известных, крупных и успешных компаний и в многочисленных приложениях.[2]

Содержание



1 Установка библиотеки

Ubuntu Linux

Достаточно стандартно выполнить команду

Код Bash

```
sudo apt-get install curl libcurl3 libcurl3-dev
```

В дальнейшем собирать исходники надо с опцией **-l**curl, например

Код Bash

```
g++ main.o -o prog -lcurl
```

Windows

1. Заходим на сайт http://curl.haxx.se/download.html. Качаем версию библиотеки для Win32 - MSVC http://curl.haxx.se/download/libcurl...2-ssl-msvc.zip

2. Распаковываем архив берем от туда все dll- файлы:

1. curllib.dll ;
2. libeay32.dll ;
3. openldap.dll ;
4. ssleay32.dll .

И папку curl (там заголовойные файлы) находящуюся в папке .\libcurl-7.19.3-win32-ssl-msvc\include

3. Так же берем curllib.lib из папки .\libcurl-7.19.3-win32-ssl-msvc\lib\Release
Если у вас **MSVC++** используем его, если **C++Builder** конвертируем файл curllib.lib в curllib-bcb.lib утилитой coff2omf.exe

Код Code

```
coff2omf curllib.lib curllib-bcb.lib
```

Теперь для подключения curl нужно написать

Код C++

```
#include "curl/curl.h"
#pragma comment(lib,"curllib.lib")            // для MSVC++
// #pragma comment(lib,"curllib-bcb.lib") // для C++Builder
```

Для работы программы также может понадобиться libsasl.dll (из OpenSSL) и возможно какие-нибудь библиотеки из MSVC++(Если у вас C++Builder).В таком случае при запуске программы из IDE она сразу же будет прекращать работу после без какой либо ошибки. Если же запустить сам exe файл то вылезит окошко указывающее на недостающую библиотеку. Требуемые dll несложно найти и скачать с интернета.

2. Основные принципы

cUrl предоставляет несколько интерфейсов:

1. Easy *(Простой режим)*
2. Multi *(Многопоточный режим)*
3. Share

Цитата:

При использовании в libcurl "простого" интерфейса вы инициализируете сеанс и получаете handle (часто упоминается как "easy handle"), который вы используете в качестве аргумента в функциях Easy интерфейса. Используйте curl_easy_init, чтобы получить handle.

После получения вы должны установить все нужные параметры в предстоящей передаче, наиболее важным среди которых является URL (передавать что-то без заданного URL невозможно). Вы можете задавать различные функции обратного вызова, которые будут вызываться из библиотеки, при получении данных и т.д. Для всего этого используется curl_easy_setopt.[2]

Список опций можно найти в документации [1]

Цитата:
После того как все настройки окончены, вы сообщаете libcurl выполнение передачи с помощью curl_easy_perform. Она проделает все операции и вернет результат своей работы типа перечисления CURLcode.

После передачи, вы можете установить новые настройки и сделать еще передачу, или, если вы уже закончили, вызовите очистку сессии curl_easy_cleanup. Если вы хотите иметь постоянное подключение, не освобождайте handle сразу, вместо этого выполните другие передачи с использованием этого же handle.

Никогда не используйте один и тот же хендл в разных потоках! (можно сделать синхронизацию, но это сведёт на нет плюсы многопоточности)[2]

3. Простой пример

Код C++

```
#include <stdio.h>

#include "curl/curl.h"
#pragma comment(lib,"curllib-bcb.lib") // Для C++Builder
//#pragma comment(lib,"curllib.lib")    // для VC++
//---------------------------------------------------------------------------
int main()
{
    CURL *curl_handle;
    CURLcode res;

    curl_handle = curl_easy_init();
    if(curl_handle)
        {
            // задаем  url адрес
            curl_easy_setopt(curl_handle, CURLOPT_URL,
"http://www.cyberforum.ru");
            // выполняем запрос
            res = curl_easy_perform(curl_handle);
                        // закрываем дескриптор curl
            curl_easy_cleanup(curl_handle);
        }

    getchar();
    return 0;
}
//---------------------------------------------------------------------------
```

В итоге в окне консоли получаем html-код страницы форума.

Код C++

```
CURL *curl_easy_init();
```

Начинает easy(простую) curl-сессию, возвращает её дескриптор, в случае неудачи NULL.
Каждому вызову такой ф-ции должен соответствовать вызов ф-ции

Код C++

```
void curl_easy_cleanup(CURL * curl);
```

для завершении работы.

Код C++
```
CURLcode curl_easy_setopt(CURL *curl, CURLoption option,
parameter);
```
Задает соответствующего поведение через установку опций, возвращает CURLE_OK (0) если успешно, в противном случае код ошибки определенный константами в файле curl/curl.h.
Опция CURLOPT_URL относится NETWORK-опциям и задает url- адрес. Этот параметр должен иметь вид согласно RFC 3986 формата: "scheme://host:port/path" *(смотри также Преобразование в URL вид.)*

Код C++
```
CURLcode curl_easy_perform(CURL *curl);
```
Выполняет отправку запроса на сервер и возвращает CURLE_OK в случае успеха, в противном случае код ошибки. [1]

4. Заголовки.

Заголовки запроса можно также установить с помощью опций, но в зависимости от версии libcurl, константы определяющие опции могут отличаться.

Код C++
```
curl_easy_setopt(curl_handle, CURLOPT_USERAGENT,"Mozilla/5.0 (Windows NT 6.1;
rv:16.0) Gecko/20100101 Firefox/16.0");
curl_easy_setopt(curl_handle, CURLOPT_ENCODING, "gzip,deflate"); // если curl
скомпилина вместе с gzip
// curl_easy_setopt(curl_handle, CURLOPT_TCP_KEEPALIVE , 1);  // не нашло такой
опции в версии 7.19.3
// curl_easy_setopt(curl_handle, CURLOPT_REFERER,"http://some.com"); //
устанавливает referer
curl_easy_setopt(curl_handle, CURLOPT_AUTOREFERER,1);// автоматически заполняет
поле referer
```
Если установить

Код C++
```
curl_easy_setopt(curl_handle, CURLOPT_HEADER, 1);
```
то заголовки ответа сервера будут отображаться в месте с html-кодом страницы (заголовок + тело)

5. Обработка ошибок.

В случае возникновения ошибки при выполнении функции curl_easy_perform() можно получить её описание с помощью:

Код C++
```
const char *curl_easy_strerror(CURLcode errornum);
```
Эта функция возвращает строку с описанием кода ошибки CURLcode указанным в аргументе:

Код C++
```
CURLcode res = curl_easy_perform(curl_handle);
if(res != CURLE_OK)
      printf( "curl_easy_perform() failed: %s\n",
```

Код C++

```
curl_easy_strerror(res) );
```

Так же есть возможность получить сообщение об ошибке из буфера ошибок, для этого нужно включить параметр CURLOPT_ERRORBUFFER с помощью ф-ции curl_easy_setopt() указать буфер.

Код C++

```
static char ErrorBuffer[CURL_ERROR_SIZE]; // размер
определяется константой curl
//...
curl_easy_setopt(curl_handle, CURLOPT_ERRORBUFFER,
errorBuffer); // указывает буфер ошибок
//...
CURLcode res = curl_easy_perform(curl_handle);
if(res != CURLE_OK)
     cout<<"Error!"<<ErrorBuffer<< endl;
```

6. Загрузка в буфер.

По умолчанию curl выводит данные в stdout т.е. в окно консоли. Для того что бы сохранить данные в отдельном буфере нужно указать этот буфер в опции CURLOPT_WRITEDATA и callback функцию которая будет записывать туда данные при их приёме с помощью опции CURLOPT_WRITEFUNCTION.

Функция должна иметь вид:

Код C++

```
size_t function( char *ptr, size_t size, size_t nmemb, void*
userdata);
```

char * ptr - указатель на принимаемые данные.
size_t size - размер принимамого блока данных
size_t nmemb - общее количество блоков данных.
void* userdata - это параметр опции CURLOPT_WRITEDATA, в который производится запись - наш буфер.

Функция должна возвращать количество обработанных байт (size*nmemb). Если это количество будет отличаться от суммы, полученной на входе вашей функции, то будет отдан сигнал об ошибке в библиотеке. Это можно использовать для прервания передачи, с возвращемым значением CURLE_WRITE_ERROR.

Функция может вернуть значение CURL_WRITEFUNC_PAUSE, которое приведет к приостановке записи в этом соединении.

Код C++

```
//---------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>

#include "curl/curl.h"
#pragma comment(lib,"curllib-bcb.lib") // Для C++Builder
//#pragma comment(lib,"curllib.lib")    // для VC++
 //---------------------------------------------------------------------------
```

Код C++

```
const size_t BUF_SIZE= 5000000;
char   wr_buf[BUF_SIZE+1];  // char*   wr_buf[BUF_SIZE+1];
size_t  wr_index=0;
 //------------------------------------------------------------------------------
static size_t write_data(char *ptr, size_t size, size_t nmemb, char* data)
{
    if(data==NULL || wr_index + size*nmemb > BUF_SIZE) return 0; // Если выход за
размеры буфера - ошибка

    memcpy( &data[wr_index], ptr, size*nmemb);// дописываем данные в конец
    wr_index+= size*nmemb;  // изменяем  текущее положение

    return size*nmemb;
}
//------------------------------------------------------------------------------
int main()
{
    CURL *curl_handle;
    CURLcode res;

    memset(wr_buf,BUF_SIZE+1,0); // заполняем буфер нулями.

    curl_handle = curl_easy_init();
    if(curl_handle)
        {
            curl_easy_setopt(curl_handle, CURLOPT_URL, "http://www.cyberforum.ru");

            curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, wr_buf);
            curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, write_data);

            CURLcode res = curl_easy_perform(curl_handle);
            if(res == CURLE_OK)
              {
                printf("%s\n Done!",wr_buf); // выводим буфер в консоль.
              }
            else printf( "curl_easy_perform() failed: %s\n", curl_easy_strerror(res) );

            curl_easy_cleanup(curl_handle);
        }

    getchar();
    return 0;
}
//------------------------------------------------------------------------------
```

В качестве буфера можно использовать контейнер или поток STL что упростит задачу.

Можно например использовать std::string.

Код C++

```
//------------------------------------------------------------------------------
```

Код C++

```cpp
#include <string>
#include <iostream>

#include "curl/curl.h"
#pragma comment(lib,"curllib-bcb.lib") // Для C++Builder
//#pragma comment(lib,"curllib.lib")    // для VC++
//---------------------------------------------------------------------------
static size_t write_data(char *ptr, size_t size, size_t nmemb, string* data)
{
  if (data)
   {
     data->append(ptr, size*nmemb);
     return size*nmemb;
   }
  else return 0;  // будет ошибка
}
//---------------------------------------------------------------------------
int main()
{
  CURL *curl_handle;
  curl_handle = curl_easy_init();

  std::string content;

  if (curl_handle)
    {
       curl_easy_setopt(curl_handle, CURLOPT_URL, "google.com");

       curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, writer);
       curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, &content);

       CURLcode res = curl_easy_perform(curl_handle);
       if (res)  std::cout << content << std::endl;
       else      std::cerr << curl_easy_strerror(res) << std::endl;

       curl_easy_cleanup(curl_handle);
     }

  getchar();
  return 0;
}
//---------------------------------------------------------------------------
```

7. Загрузка в файл

Следующий пример показывает как можно сохранить тело ответа сервера в один в файл, а заголовок в другой.

Код C++

Код C++

```cpp
#include <stdio.h>
#include <stdlib.h>

#include "curl/curl.h"
#pragma comment(lib,"curllib-bcb.lib")
//---------------------------------------------------------------------------
size_t write_data( char *ptr, size_t size, size_t nmemb, FILE* data)
{
    return fwrite(ptr, size, nmemb, data);
}
//---------------------------------------------------------------------------
int main()
{
    // Открываем файлы для заголовка и тела

    const char *headerfilename= "head.txt";
    const char *bodyfilename  = "body.html";

    FILE *headerfile= fopen(headerfilename,"w");
    if (headerfile == NULL) return -1;

    FILE *bodyfile =  fopen(bodyfilename,"w");
    if (bodyfile == NULL)  return -1;

    // Выполняем  запрос

    CURL *curl_handle = curl_easy_init();

    if(curl_handle)
        {
            curl_easy_setopt(curl_handle, CURLOPT_URL, "http://www.cyberforum.ru");

            // сохраняем тело
            curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, bodyfile);
            curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, write_data);

            // сохраняем заголовок
            curl_easy_setopt(curl_handle, CURLOPT_WRITEHEADER, headerfile);

            CURLcode res = curl_easy_perform(curl_handle);
            if(res != CURLE_OK)
                 printf( "curl_easy_perform() failed: %s\n", curl_easy_strerror(res) );
            curl_easy_cleanup(curl_handle);
        }

    puts("\nDone!");
    getchar();
    return 0;
}
```

Код C++

```
50  //---------------------------------------------------------------------------
```

8. Перенаправление (редирикт).

Для автоматического перехода на перенаправляемую страницу необходимо установить опцию CURLOPT_FOLLOWLOCATION в **1**

Код C++

```
1  curl_easy_setopt(curl_handle, CURLOPT_FOLLOWLOCATION, 1);
```

Для того что бы ограничить количество перенаправлений нужно установить опцию CURLOPT_MAXREDIRS её параметр указывает их максимальное количество

Код C++

```
1  curl_easy_setopt(curl_handle, CURLOPT_MAXREDIRS, 10); //  останавливаться после 10-ого
   редиректа
```

9. Cookie

Далее описание опций для работы с Cookie

CURLOPT_COOKIE

Код C++

```
1  curl_easy_setopt(curl_handle, CURLOPT_COOKIE, cookiestring);
```

Параметр -указатель на строку в стиле Си для установки cookies в http-запросе .
Формат строки должен быть вида name=contents, где name имя cookie, а contents- её содержание.

Используется, когда вы хотите указать точное содержание cookie-заголовоков для отправки на сервер.

Если нужно передать несколько cookie, то строка должна выглядеть как "name1 = content1; name2 = content2;" и т. д.

CURLOPT_COOKIEFILE

Код C++

```
1  curl_easy_setopt(curl_handle, CURLOPT_COOKIEFILE,
   cookiefilename);
```

Принимает указатель на строку в стиле Си, которой хранится путь к файлу, который содержит cookies. Cookies должны хранится в формате куков Netscape/Mozilla или в обычном HTTP-стиле заголовков (Set-Cookie: ...) помещенных в файл.

Если указать несуществующий файл или пустую строку (**""**), то это разрешит libcurl полученные использовать cookie в следующих запросах для данного дескриптора curl

Можно несколько раз устанавливать эту опцию для загрузки нескольких файлов с куками.

CURLOPT_COOKIEJAR

Код C++

```
1  curl_easy_setopt(curl_handle, CURLOPT_COOKIEJAR,
   cookiefilename);
```

Запишет все известные cookies в указанный файл. Если cookies нет файл не будет создан. Если указать **"-"** вместо имени файла cookies будут выведены в окно консоли (stdout), и будет разрешено использование cookies для этой сессии

Если файл не может быть создан или сохранен libcurl **не** выдаст ошибку.
Использование опций CURLOPT_VERBOSE или CURLOPT_DEBUGFUNCTION вызовет вывод предупреждения.

CURLOPT_COOKIESESSION

Код C++

```
1  curl_easy_setopt(curl_handle, CURLOPT_COOKIESESSION, 1);
```

Установка в **1** укажет текущему сеансу начать новую "сессию" cookies. Это заставит libcurl проигнорировать все "сессионные" cookies, которые она должна была бы загрузить, полученные из предыдущей сессии. По умолчанию, libcurl всегда сохраняет и загружает все cookies, вне зависимости от того, являются ли они "сессионными" или нет. "Сессионные" cookies - это cookies без срока истечения, которые должны существовать только для текущей "сессии".

CURLOPT_COOKIELIST

Код C++

```
curl_easy_setopt(curl_handle, CURLOPT_COOKIELIST, "ALL"); // Чистка cookies
```

Устанавливает cookies.

Если задать параметр **"ALL"** то все cookies будут очищены , если **"FLUSH"** будут сохранены в файл указанный в опции CURLOPT_COOKIEJAR

Для вывода информации о cookie можно воспользоваться функцией из примера *cookie_interface.c*

Код C++

```
static void print_cookies(CURL *curl)
{
    CURLcode res;
    struct curl_slist *cookies;
    struct curl_slist *nc;
    int i;

    printf("Cookies, curl knows:\n");
    res = curl_easy_getinfo(curl, CURLINFO_COOKIELIST, &cookies);
    if (res != CURLE_OK)
        {
         fprintf(stderr, "Curl curl_easy_getinfo failed: %s\n",
                         curl_easy_strerror(res ));
         exit(1);
        }
    nc= cookies, i = 1;
    while (nc)
        {
         printf("[%d]: %s\n", i, nc->data);
         nc = nc->next;
         i++;
        }
    if (i == 1)  printf("(none)\n");

    curl_slist_free_all(cookies);
}
```

10. Преобразование в URL вид.

Иногда возникает необходимость передавать GET/POST запросы параметры которые содержать в себе символы требующие "экранирования" (кириллица, символ "@").
К примеру необходимо выполнить поиск в яндексе по слову "Программирование", если посмотреть в строку браузера, то там это будет выглядеть так:

Код HTML5

```
http://yandex.ua/yandsearch?text=Программирование
```

Слово "Программирование" должно экранироваться при запросе:

Код HTML5

Код HTML5

```
http://yandex.ua/yandsearch?text=%CF%F0%EE%E3%F0%E0%EC%EC%E8%F0%EE%E2%E0%ED%E8%E5
```

Для этих целей в curl предусмотрена функция

Код C++

```
char* curl_easy_escape( CURL* curl , char* url , int length );
```

Цитата:

Эта функция преобразует входной строку в закодированную для URL строку и возвращает ее в качестве новой выделенной строки. Все входные символы, кроме a-z, A-Z или 0-9 преобразуются в их "замаскированные" версии (%NN, где NN - двузначное шестнадцатеричное число).

Если длина аргумента имеет значение 0 (ноль), curl_easy_escape использует strlen() на входной строке, чтобы вычислить размер.

Вы должны освободить полученную строку с помощью **curl_free**, после окончания работы с ней.[2]

Код C++

```
void curl_free ( char* ptr );
```

Цитата:

curl_free освобождает память, которая была выделена внутри функций curl. Необходимо использовать curl_free() вместо free(), чтобы избежать ошибок, которые могут возникнуть по причине возможных различий при управлении памятью в вашем приложении и curl.[2]

Код для рассматриваемого примера

Код C++

```
//---------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>

#include <string>
#include <iostream>

#include "curl/curl.h"
#pragma comment(lib,"curllib-bcb.lib") // Для C++Builder
//#pragma comment(lib,"curllib.lib")     // для VC++
//---------------------------------------------------------------------------
int main()
{
    CURL *curl_handle;
    CURLcode res;

    curl_handle = curl_easy_init();

    if(curl)
        {
            char ru_text[]= "Программирование";
                        // Преобразование в URL- вид
            char* esc_text= curl_easy_escape( curl_handle, ru_text, 0);
            if(!esc_text)
                                {
                                    std::cout<<"can not convert string to URL" <<std::endl;
                                    curl_easy_cleanup(curl_handle);
                                    getchar();
```

Код C++

```
                                return 1;
                            }

            std::string url=  "http://yandex.ua/yandsearch?text=";
                url+= esc_text;

            curl_free(esc_text);

                // задаем  url адрес
             curl_easy_setopt(curl_handle, CURLOPT_URL, url.c_str() );
                // разрешаем перенаправление
             curl_easy_setopt(curl_handle, CURLOPT_FOLLOWLOCATION, 1);
             // выполняем запрос
             res = curl_easy_perform(curl_handle);
                        curl_easy_cleanup(curl_handle);
          }
     getchar();
     return 0;
}
//---------------------------------------------------------------------------
```

Для удобства использования с std::string можно написать такую функцию

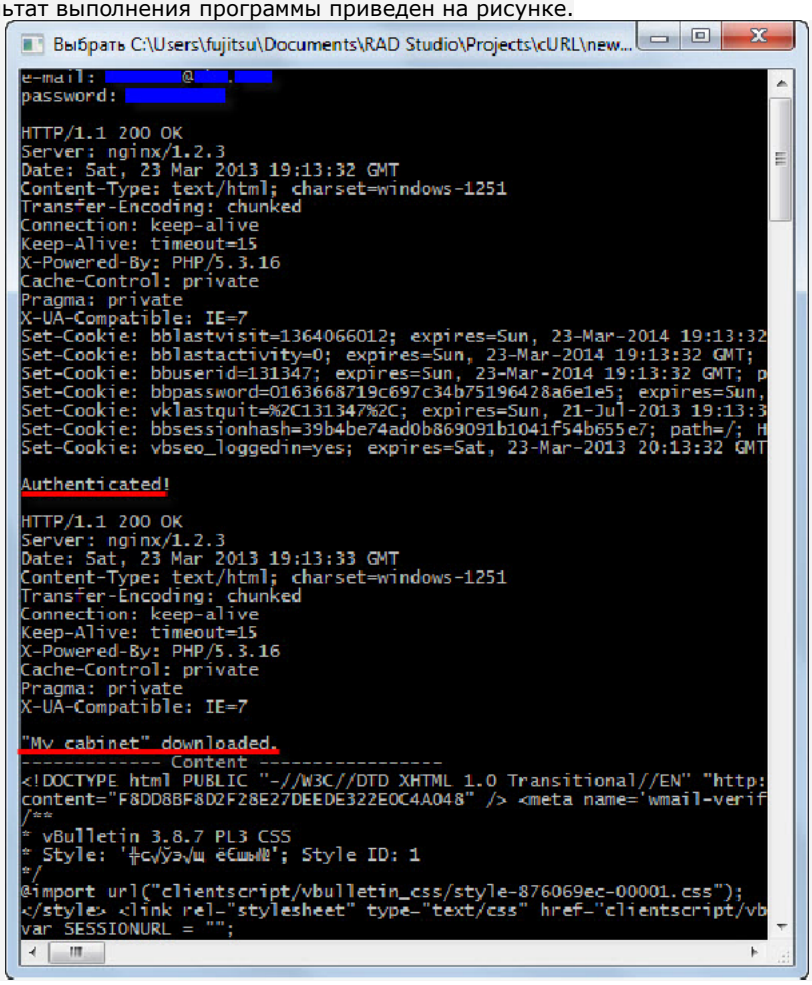
Код C++

```
//---------------------------------------------------------------------------
std::string escape(CURL *curl_handle, const std::string& text)
{
    std::string result;
    char* esc_text= curl_easy_escape(curl_handle, text.c_str(), text.length());
    if(!esc_text) throw std::runtime_error("Can not convert string to URL");

    result = esc_text;
    curl_free(esc_text);

    return result;
}
//---------------------------------------------------------------------------
```

11. POST-запрос для авторизации на форуме.

В качестве примера приведу авторизацию на cyberforum.ru с последующим переходом в "Мой кабинет", для работы примера вам необходимо указать свой e-mail и пароль.

Результат выполнения программы приведен на рисунке.



Код C++

```
#include <stdio.h>
#include <stdlib.h>

#include <string>
#include <iostream>

#include "curl/curl.h"
#pragma comment(lib,"curllib-bcb.lib") // Для C++Builder
//#pragma comment(lib,"curllib.lib")     // для VC++
//---------------------------------------------------------------------------
std::string escape(CURL *curl_handle, const std::string& text)
{
        std::string result;
        char* esc_text= curl_easy_escape(curl_handle, text.c_str(), text.length());
        if(!esc_text) throw std::runtime_error("Can not convert string to URL");

        result = esc_text;
        curl_free(esc_text);
```

Код C++

```
19
20          return result;
21  }
22  //------------------------------------------------------------------------
23  static size_t write_data(char *ptr, size_t size, size_t nmemb, std::string* data)
24  {
25      if (data)
26       {
27           data->append(ptr, size*nmemb);
28           return size*nmemb;
29       }
30      else return 0;  // будет ошибка
31  }
32  //------------------------------------------------------------------------
33  static size_t write_head(char *ptr, size_t size, size_t nmemb, std::ostream* stream)
34  {
35      (*stream)<< std::string(ptr, size*nmemb);
36      return size*nmemb;
37  }
38  //------------------------------------------------------------------------
39  int main()
40  {
41      std::string content;
42
43      /* Пользовательские данные */
44      std::string url_dologin= "http://www.cyberforum.ru/login.php?do=login";// страница авторизации
45      std::string user_name;  //  e-mail
46      std::string password;   //  пароль
47
48      std::cout<<"e-mail: ";              getline(std::cin,user_name);
49      std::cout<<password<<"password: ";  getline(std::cin,password);
50      std::cout<<std::endl;
51
52   std::string url_user= "http://www.cyberforum.ru/usercp.php"; // Мой кабинет
53
54      CURL *curl_handle = curl_easy_init();
55      if(curl_handle)
56          {
57              /* Формирование  запроса на основе пользовательских данных */
58              std::string post_data;
59              post_data+= "vb_login_username="  + escape(curl_handle, user_name);
60              post_data+= "&cookieuser=1";
61              post_data+= "&vb_login_password=" + escape(curl_handle, password);
62              post_data+= "&s=&securitytoken=guest";
63              post_data+= "&do=login";
64              post_data+= "&vb_login_md5password=";
65              post_data+= "&vb_login_md5password_utf=";
66
67              curl_easy_setopt(curl_handle, CURLOPT_URL, url_dologin.c_str() );
```

Код C++

```
68
69              //  сохраняем html код страницы в строку content
70              curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, write_data);
71              curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA,     &content);
72
73              // Загловок ответа сервера выводим в консоль
74              curl_easy_setopt(curl_handle, CURLOPT_HEADERFUNCTION, write_head);
75              curl_easy_setopt(curl_handle, CURLOPT_WRITEHEADER,    &std::cout);
76
77              // авто перенаправление
78              curl_easy_setopt(curl_handle, CURLOPT_FOLLOWLOCATION, 1);
79              // max 5-ть перенаправлений
80              curl_easy_setopt(curl_handle, CURLOPT_MAXREDIRS, 5);
81              // разрешаем использовать куки
82              curl_easy_setopt(curl_handle, CURLOPT_COOKIEFILE,"");
83
84              /* POST- запрос с авторизацией ( просходит получение кукисов ) */
85              curl_easy_setopt(curl_handle, CURLOPT_POSTFIELDS, post_data.c_str() );
86              curl_easy_setopt(curl_handle, CURLOPT_POSTFIELDSIZE, post_data.length() );
87
88              CURLcode res = curl_easy_perform(curl_handle);
89              if(res != CURLE_OK)
90                  {
91                      std::cout<< curl_easy_strerror(res) << std::endl;
92                      getchar();
93                      return 0;
94                  }
95              // Проверяем успешно ли авторизировались
96              if( content.find("Спасибо, что зашли") != std::string::npos )
97                      std::cout << "Authenticated!" <<std::endl<<std::endl;
98              else
99                  {
100                     std::cout<< "Non-Authenticated!" <<std::endl;
101                     curl_easy_cleanup(curl_handle);
102                     getchar();
103                     return 0;
104                 }
105             /* GET- запрос для перехода в "Мой кабинет" форума */
106             content.clear();
107             // меняем POST-режим на GET
108             curl_easy_setopt(curl_handle, CURLOPT_POST, 0);
109             curl_easy_setopt(curl_handle, CURLOPT_HTTPGET, 1);
110             // меняем url
111             curl_easy_setopt(curl_handle, CURLOPT_URL, url_user.c_str() );
112
113             res = curl_easy_perform(curl_handle);
114             if(res != CURLE_OK)  std::cout<< curl_easy_strerror(res) <<std::endl;
115
116             // Проверяем получили то что ожидали
```

Код C++

```
117             if( content.find("Мой кабинет") != std::string::npos)
118                         std::cout << "\"My cabinet\" downloaded."  <<std::endl
119                                                 << "------------- Content -----------------" <<std::endl
120                                                 << content <<std::endl;
121             else  std::cout << "Is not \"My cabinet\" page" <<std::endl;
122
123             curl_easy_cleanup(curl_handle);
124         }
125
126     getchar();
127     return 0;
128 }
129 //---------------------------------------------------------------------------
```

Основные моменты:

1. Для успешной авторизации и перехода по страницам нужно:

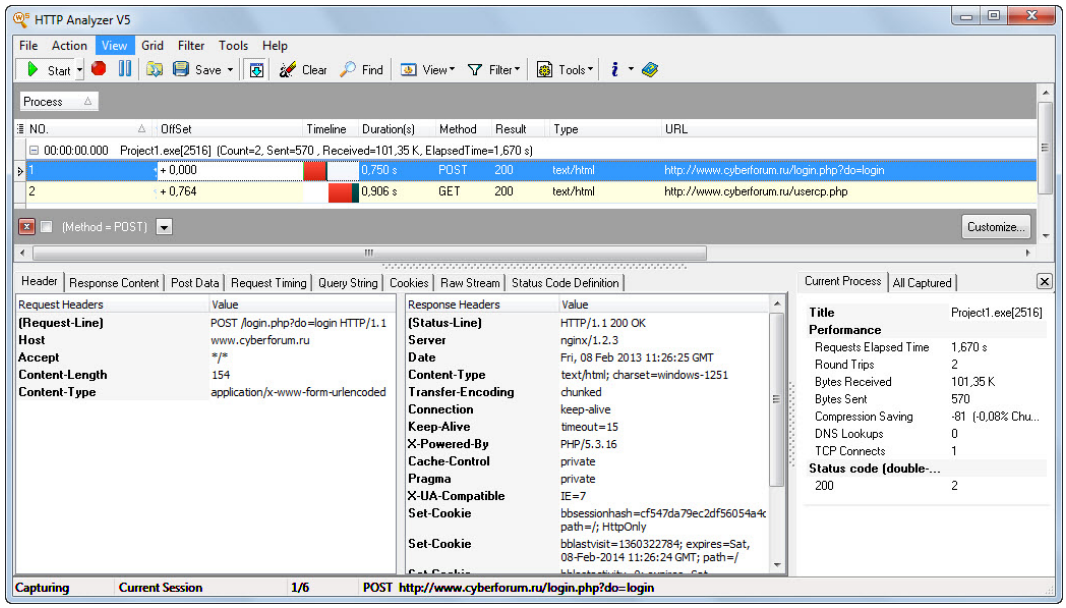
- разрешить использовать полученные cookie;
- включить автоматический переход по перенаправлениям.

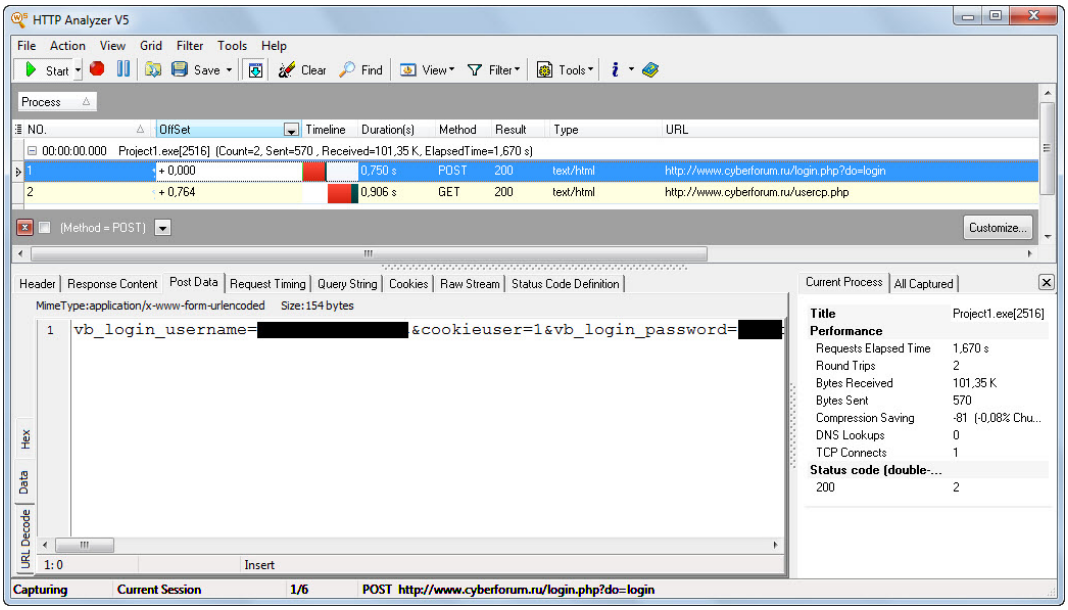
2. По умолчанию libcurl использует GET "режим" отправки запросов, после установки опции CURLOPT_POSTFIELDSIZE он переключается на POST, поэтому для осуществление последующего GET нужно переключить режим на GET:

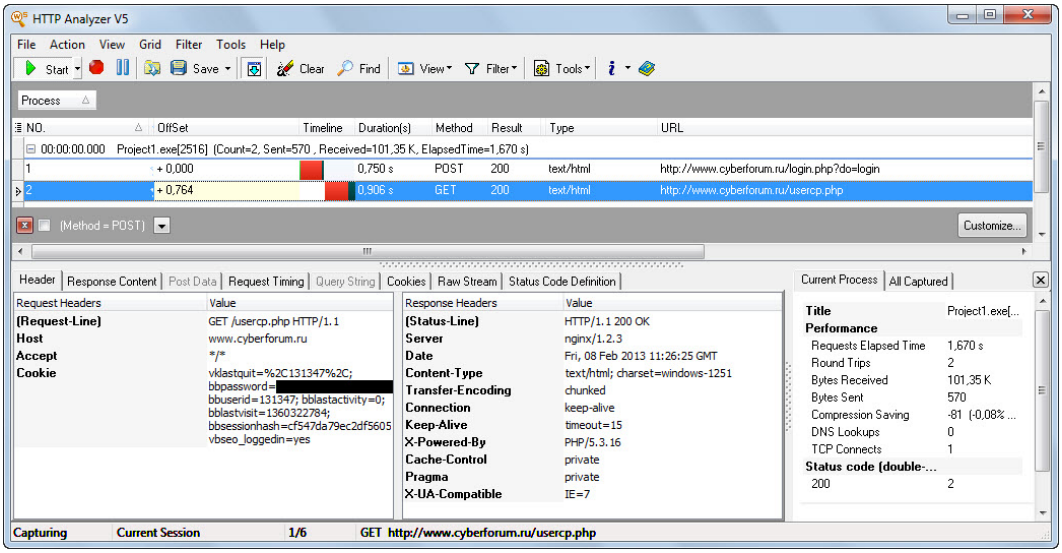
Код C++

```
1 curl_easy_setopt(curl_handle, CURLOPT_POST, 0); // выключаем POST
2 curl_easy_setopt(curl_handle, CURLOPT_HTTPGET, 1);// включаем GET
```

4. Для анализа того что отправляет/принимает ваше приложение удобно использовать снифер. Я пользуюсь к примеру HTTP Analyzer V5.







12. Получение страниц в сжатом виде.

Код C++

```
1  //---------------------------------------------------------------------------
2  #include <string>
3  #include <iostream>
4
5  #include "curl/curl.h"
6  #pragma comment(lib,"curllib-bcb.lib") // Для C++Builder
7  //#pragma comment(lib,"curllib.lib")    // для VC++
8  //---------------------------------------------------------------------------
9  int main()
10 {
11     std::string url= "http://www.google.ru/";
12
13     CURL *curl_handle = curl_easy_init();
14     if(curl_handle)
15         {
16             curl_easy_setopt(curl_handle, CURLOPT_URL, url.c_str() );
```

Код C++

```
            curl_easy_setopt(curl_handle, CURLOPT_ENCODING, "gzip,deflate"); // Принудительно ставим gzip

            CURLcode res = curl_easy_perform(curl_handle);
            if(res != CURLE_OK)
                 std::cout<<"Error #"<<res<<" "<<curl_easy_strerror(res) <<std::endl;

            curl_easy_cleanup(curl_handle);
        }

    std::cout<<std::endl<<"Done!";
    getchar();
    return 0;
}
//---------------------------------------------------------------------------
```

13. Передача с использованием HTTPs.

Для работы с по протоколу HTTPs необходимо что бы libcurl была собрана с поддержкой SSL, а также необходимы соответствующие dll-ки *(из OpenSSL)*
Если библиотека собрана без SSL, то ф-ция curl_easy_perform() вернет код CURLE_UNSUPPORTED_PROTOCOL (1) "Unsupported protocol"

Далее пример простого http**s**- запроса :

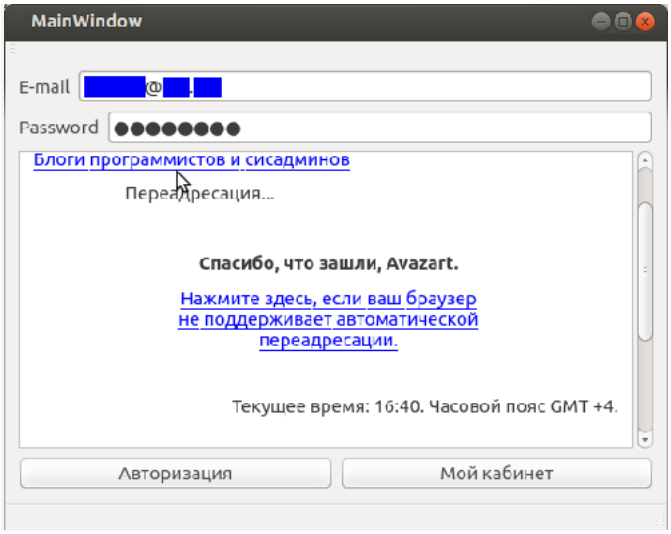
Код C++

```
//---------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>

#include <string>
#include <iostream>

#include "curl/curl.h"
#pragma comment(lib,"curllib-bcb.lib") // Для C++Builder
//#pragma comment(lib,"curllib.lib")    // для VC++
//---------------------------------------------------------------------------
size_t write_data( char *ptr, size_t size, size_t nmemb, FILE* data)
{
    return fwrite(ptr, size, nmemb, data);
}
//---------------------------------------------------------------------------
int main()
{
    // Открываем файлы для заголовка и тела
    std::string body_filename  = "body.html";
    FILE *body_file =  fopen(body_filename.c_str(),"w");
    if (body_file == NULL)  return -1;

    std::string url= "https://my.webmoney.ru/login.aspx";

    CURL *curl_handle = curl_easy_init();
```

Код C++

```
27      if(curl)
28          {
29              // сохранение в файл html-страницу
30              curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, body_file);
31              curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, write_data);
32              // заголовки ответа выводим в консоль
33              curl_easy_setopt(curl_handle, CURLOPT_WRITEHEADER, stdout);
34
35                 /* HTTPs  Запрос */
36              curl_easy_setopt(curl_handle, CURLOPT_URL, url.c_str() );
37
38               // не проверять SSL сертификат
39              curl_easy_setopt(curl_handle, CURLOPT_SSL_VERIFYPEER, 0);
40              // не проверять Host SSL сертификата
41              curl_easy_setopt(curl, CURLOPT_SSL_VERIFYHOST, 0);
42
43              CURLcode res = curl_easy_perform(curl_handle);
44              if(res != CURLE_OK)
45                   std::cout<<"Error #"<<res<<" "<<curl_easy_strerror(res) <<std::endl;
46
47               curl_easy_cleanup(curl_handle);
48          }
49
50      std::cout<<std::endl<<"Done!";
51      getchar();
52      return 0;
53  }
54  //---------------------------------------------------------------------------
```

Разультат запроса будет сохранен в файле body.html, заголовок ответа сервера будет выведен на экран.



Исходники:

HTTP, POST, авторизация на cyberforum (curl v7.19.3-ssl, C++Builder XE3).rar
HTTP, GET, (curl v7.22.0, g++ 4.6.3, Makefile, Ubuntu x32).rar
HTTP, POST, авторизация на cyberforum(curl v7.22.0, Qt5, Ubuntu x32).zip

Литература:

1. http://curl.haxx.se/
2. http://ru.libcurl.wikia.com/wiki/Lib...B8%D0%BA%D0%B8

Темы :

1. Скачивание из интернета
2. Builder и curl
3. Замена строки
4. Опять кодировки
5. Работа с CURL