转载地址：

http://blog.chinaunix.net/u/17660/showart_1822514.html

2 LibCurl 编程

2.1LibCurl 编程流程

在基于 LibCurl 的程序里，主要采用 callback function （回 调函数）的形式完成传输任务，用户在启动传输前设置好各类参数和回调函数，当满足条件时 libcurl 将调用用户的回调函数实现特定功能。下面是利用 libcurl 完成传输任务的流程：

1. 调用 curl_global_init()初始化 libcurl
2. 调用 curl_easy_init()函数得到 easy interface 型指针
3. 调用 curl_easy_setopt 设置传输选项
4. 根据 curl_easy_setopt 设置的传输选项，实现回调函数以完成用户特定任务
5. 调用 curl_easy_perform（）函数完成传输任务
6. 调用 curl_easy_cleanup（）释放内存

在整过过程中设置 curl_easy_setopt()参数是最关键的，几乎所有的 libcurl 程序都要使用它。

2.2 重要函数

1. CURLcode curl_global_init(long flags);

描述：

这个函数只能用一次。(其实在调用 curl_global_cleanup 函数后仍然可再用)

如果这个函数在 curl_easy_init 函数调用时还没调用，它讲由 libcurl 库自动完成。

参数：flags

CURL_GLOBAL_ALL　　//初始化所有的可能的调用。

CURL_GLOBAL_SSL　　//初始化支持 安全套接字层。

CURL_GLOBAL_WIN32　　//初始化 win32 套接字库。

CURL_GLOBAL_NOTHING　　//没有额外的初始化。

2 void curl_global_cleanup(void);

描述：在结束 libcurl 使用的时候，用来对 curl_global_init 做的工作清理。类似于 close 的函数。

3 char *curl_version( );

描述：打印当前 libcurl 库的版本。

4 CURL *curl_easy_init( );

描述：

curl_easy_init 用来初始化一个 CURL 的指针(有些像返回 FILE 类型的指针一样). 相应的在调用结束时要用 curl_easy_cleanup 函数清理.

一般 curl_easy_init 意味着一个会话的开始. 它的返回值一般都用在 easy 系列的函数中.

5 void curl_easy_cleanup(CURL *handle);

描述：

这个调用用来结束一个会话.与 curl_easy_init 配合着用.

参数：

CURL 类型的指针.

6 CURLcode curl_easy_setopt(CURL *handle, CURLoption option, parameter);

描述：这个函数最重要了.几乎所有的 curl 程序都要频繁的使用它.它告诉 curl 库.程序将有如何的行为. 比如要查看 一个网页的 html 代码等.(这个函数有些像 ioctl 函数)参数：

1 CURL 类型的指针

2 各种 CURLoption 类型的选项.(都在 curl.h 库 里有定义,man 也可 以查看

到)

3 parameter 这个参数 既可以是个函数的指针,也可以是某个 对象的指针,也可以是个 long 型的变量.它用什么这取决于第二个参数.

CURLoption 这个参数的取值很多.具体的可以查看 man 手册.

7 CURLcode curl_easy_perform(CURL *handle);描述:这个函数在初始化 CURL 类型的指针 以及 curl_easy_setopt 完成后调用. 就像字面的意思所说 perform 就 像是个舞台.让我们设置的

option 运作起来.参数:

CURL 类型的指针.

3.3curl_easy_setopt 函数介绍

本节主要介绍 curl_easy_setopt 中跟 http 相 关的参数。注意本节的阐述都是以 libcurl 作为主体，其它为客体来阐述 的。

1. CURLOPT_URL

设置访问 URL

2. CURLOPT_WRITEFUNCTION，CURLOPT_WRITEDATA

回调函数原型为：**size_t function(void *ptr, size_t size, size_t nmemb, void * stream);** 函 数将在 libcurl 接收到数据后被调用，因此函数多做数据保存的功能，如处理下载文件。CURLOPT_WRITEDATA 用于表明

CURLOPT_WRITEFUNCTION 函 数中的 stream 指针的来源。

3. CURLOPT_HEADERFUNCTION，CURLOPT_HEADERDATA

回调函数原型为 size_t function(void *ptr, size_t size,size_t nmemb, void *stream); libcurl 一旦接收到 http 头部数据后将调用该函数。

CURLOPT_WRITEDATA 传递指针给 libcurl，该指针表明

CURLOPT_HEADERFUNCTION 函数的 stream 指针的来源。

4. CURLOPT_READFUNCTION CURLOPT_READDATA

libCurl 需要读取数据 传递给远程主机时将调用

CURLOPT_READFUNCTION 指定的函数，函数原型是：size_t function (void *ptr, size_t size, size_t nmemb,void *stream).

CURLOPT_READDATA 表明 CURLOPT_READFUNCTION 函数原型中的 stream 指针来源。

5. CURLOPT_NOPROGRESS，CURLOPT_PROGRESSFUNCTION，CURLOPT_PROGRESSDATA

跟数据传输进度相关的参数。CURLOPT_PROGRESSFUNCTION 指定的函数正常情况下每秒被 libcurl 调用一次，为了使 CURLOPT_PROGRESSFUNCTION 被调用，CURLOPT_NOPROGRESS 必须 被设置为 false，CURLOPT_PROGRESSDATA 指 定的参数将作为 CURLOPT_PROGRESSFUNCTION 指定函数的第一个参数

6. CURLOPT_TIMEOUT，CURLOPT_CONNECTIONTIMEOUT:

CURLOPT_TIMEOUT 由于设置传输时间，

CURLOPT_CONNECTIONTIMEOUT 设置连接等待时间

7. CURLOPT_FOLLOWLOCATION

设置重定位 URL

CURLOPT_RANGE: CURLOPT_RESUME_FROM:

断点续传相关设置。CURLOPT_RANGE 指定 char *参数传递给 libcurl，用于指明 http 域的 RANGE 头域，例如：

表示头 500 个字节：bytes=0-499

表示第二个 500 字节：bytes=500-999

表示最后 500 个字节：bytes=-500

表示 500 字节以后的范围：bytes=500-

第一个和最后一个字节：bytes=0-0,-1

同时指定几个范围：bytes=500-600,601-999

CURLOPT_RESUME_FROM 传 递一个 long 参数给 libcurl，指定你希望开始传递的
偏移量。

3.4curl_easy_perform 函数说明（error 状 态码）

该函数完成 curl_easy_setopt 指定的所有选项，本节重点介绍 curl_easy_perform 的返回值。返回 0 意味一 切 ok，非 0 代表错误发生。主要错误码说明：

1. CURLE_OK
任务完成一切都好
2 CURLE_UNSUPPORTED_PROTOCOL
不支持的协议，由 URL 的头部指定
3 CURLE_COULDNT_CONNECT
不能连接到 remote 主机或者代理
4 CURLE_REMOTE_ACCESS_DENIED
访问被拒绝
5 CURLE_HTTP_RETURNED_ERROR
Http 返回错误
6 CURLE_READ_ERROR
读本地文件错误

3.1 获取 html 网页

```
#include <stdio.h>
#include <curl/curl.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    CURL *curl;             //定义 CURL 类型的指针
     CURLcode res;          //定义 CURLcode 类型的变量，保存返回状态码
    if(argc!=2)
    {
```

```
        printf("Usage : file <url>;\n");
        exit(1);
    }

    curl = curl_easy_init();        //初始化一个 CURL 类型的指针
    if(curl!=NULL)
    {
        //设置 curl 选项. 其中 CURLOPT_URL 是让用户指 定 url. argv[1]中存放的命令行传进来的网址
        curl_easy_setopt(curl, CURLOPT_URL, argv[1]);
        //调用 curl_easy_perform 执行我们的设置.并进行相关的操作. 在这 里只在屏幕上显示出来.
        res = curl_easy_perform(curl);
        //清除 curl 操作.
        curl_easy_cleanup(curl);
    }
    return 0;
}
```

编译 **gcc get_http.c -o get_http -lcurl**

./ get_http www.baidu.com

3.2 网页下载保存实例

```
// 采用 CURLOPT_WRITEFUNCTION 实现网页下载保存功能
#include <stdio.h>;
#include <stdlib.h>;
#include <unistd.h>;

#include <curl/curl.h>;
#include <curl/types.h>;
#include <curl/easy.h>;

FILE *fp;  //定义 FILE 类型指针
//这个函数是为了符合 CURLOPT_WRITEFUNCTION 而构造的
//完成数据保存功能
size_t write_data(void *ptr, size_t size, size_t nmemb, void *stream)
{
    int written = fwrite(ptr, size, nmemb, (FILE *)fp);
    return written;
}

int main(int argc, char *argv[])
{
    CURL *curl;
```

```
    curl_global_init(CURL_GLOBAL_ALL);
    curl=curl_easy_init();
    curl_easy_setopt(curl, CURLOPT_URL, argv[1]);

    if((fp=fopen(argv[2],"w"))==NULL)
    {
       curl_easy_cleanup(curl);
       exit(1);
    }
////CURLOPT_WRITEFUNCTION 将后继的动作交给 write_data 函数处理
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_data);
    curl_easy_perform(curl);
    curl_easy_cleanup(curl);
    exit(0);
}
```

编译 **gcc save_http.c -o save_http –lcurl**

./ save_http www.baidu.com /tmp/baidu

3.3 进度条实例??显 示文件下载进度

```
// 采用 CURLOPT_NOPROGRESS， CURLOPT_PROGRESSFUNCTION
   CURLOPT_PROGRESSDATA 实现文件传输进度提示功能
//函数采用了 gtk 库，故编译时需指定 gtk 库
//函数启动专门的线程用于显示 gtk 进度条 bar
#include <stdio.h>
#include <gtk/gtk.h>
#include <curl/curl.h>
#include <curl/types.h> /* new for v7 */
#include <curl/easy.h> /* new for v7 */

GtkWidget *Bar;
////这个函数是为了符合 CURLOPT_WRITEFUNCTION 而构造的
//完成数据保存功能
size_t my_write_func(void *ptr, size_t size, size_t nmemb, FILE *stream)
{
  return fwrite(ptr, size, nmemb, stream);
}
//这个函数是为了符合 CURLOPT_READFUNCTION 而构造的
//数据上传时使用
size_t my_read_func(void *ptr, size_t size, size_t nmemb, FILE *stream)
{
  return fread(ptr, size, nmemb, stream);
}
```

```
    //这个函数是为了符合 CURLOPT_PROGRESSFUNCTION 而构造的
    //显示文件传输进度，t 代表文件大小，d 代表传  输已经完成部分
    int my_progress_func(GtkWidget *bar,
                         double t, /* dltotal */
                         double d, /* dlnow */
                         double ultotal,
                         double ulnow)
    {
    /*   printf("%d / %d (%g %%)\n", d, t, d*100.0/t);*/
      gdk_threads_enter();
      gtk_progress_set_value(GTK_PROGRESS(bar), d*100.0/t);
      gdk_threads_leave();
      return 0;
    }

    void *my_thread(void *ptr)
    {
      CURL *curl;
      CURLcode res;
      FILE *outfile;
      gchar *url = ptr;

      curl = curl_easy_init();
      if(curl)
      {
        outfile = fopen("test.curl", "w");

        curl_easy_setopt(curl, CURLOPT_URL, url);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, outfile);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, my_write_func);
        curl_easy_setopt(curl, CURLOPT_READFUNCTION, my_read_func);
        curl_easy_setopt(curl, CURLOPT_NOPROGRESS, 0L);
        curl_easy_setopt(curl, CURLOPT_PROGRESSFUNCTION,
my_progress_func);
        curl_easy_setopt(curl, CURLOPT_PROGRESSDATA, Bar);

        res = curl_easy_perform(curl);

        fclose(outfile);
        /* always cleanup */
        curl_easy_cleanup(curl);
      }

      return NULL;
```

```
}

int main(int argc, char **argv)
{
  GtkWidget *Window, *Frame, *Frame2;
  GtkAdjustment *adj;

  /* Must initialize libcurl before any threads are started */
  curl_global_init(CURL_GLOBAL_ALL);

  /* Init thread */
  g_thread_init(NULL);

  gtk_init(&argc, &argv);
  Window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  Frame = gtk_frame_new(NULL);
  gtk_frame_set_shadow_type(GTK_FRAME(Frame), GTK_SHADOW_OUT);
  gtk_container_add(GTK_CONTAINER(Window), Frame);
  Frame2 = gtk_frame_new(NULL);
  gtk_frame_set_shadow_type(GTK_FRAME(Frame2), GTK_SHADOW_IN);
  gtk_container_add(GTK_CONTAINER(Frame), Frame2);
  gtk_container_set_border_width(GTK_CONTAINER(Frame2), 5);
  adj = (GtkAdjustment*)gtk_adjustment_new(0, 0, 100, 0, 0, 0);
  Bar = gtk_progress_bar_new_with_adjustment(adj);
  gtk_container_add(GTK_CONTAINER(Frame2), Bar);
  gtk_widget_show_all(Window);

  if (!g_thread_create(&my_thread, argv[1], FALSE, NULL) != 0)
    g_warning("can't create the thread");


  gdk_threads_enter();
  gtk_main();
  gdk_threads_leave();
  return 0;
}
```

编译 **export PKG_CONFIG_PATH=/usr/lib/pkgconfig/**

gcc progress.c －o progress ` pkg-config --libs －cflags gtk+-2..0` -lcurl －lgthread-2.0

./ progress http://software.sky-union.cn/index.asp

3.4 断点续传实例

```
//采用 CURLOPT_RESUME_FROM_LARGE 实现文件断点续传功能
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>

#include <curl/curl.h>
//这个函数为 CURLOPT_HEADERFUNCTION 参数构造
/*  从 http 头部获取文件 size*/
size_t getcontentlengthfunc(void *ptr, size_t size, size_t nmemb, void *stream) {
        int r;
        long len = 0;

        /* _snscanf() is Win32 specific */
        // r = _snscanf(ptr, size * nmemb, "Content-Length: %ld\n", &len);
 r = sscanf(ptr, "Content-Length: %ld\n", &len);
        if (r) /* Microsoft: we don't read the specs */
                *((long *) stream) = len;

        return size * nmemb;
}

/*  保存下载文件  */
size_t wirtefunc(void *ptr, size_t size, size_t nmemb, void *stream)
{
         return fwrite(ptr, size, nmemb, stream);
}

/*读取上传文件  */
size_t readfunc(void *ptr, size_t size, size_t nmemb, void *stream)
{
        FILE *f = stream;
        size_t n;

        if (ferror(f))
                return CURL_READFUNC_ABORT;

        n = fread(ptr, size, nmemb, f) * size;

        return n;
}

//  下载  或者上传文件函数
int download(CURL *curlhandle, const char * remotepath, const char * localpath,
            long timeout, long tries)
```

```
    {
        FILE *f;
        curl_off_t local_file_len = -1 ;
        long filesize =0 ;

        CURLcode r = CURLE_GOT_NOTHING;
        int c;
      struct stat file_info;
      int use_resume = 0;
      /*  得到本地文件大小  */
      //if(access(localpath,F_OK) ==0)

       if(stat(localpath, &file_info) == 0)
        {
         local_file_len =   file_info.st_size;
         use_resume   = 1;
        }
      //采用追加方式打开文件，便于实现文件断点续传工作
        f = fopen(localpath, "ab+");
        if (f == NULL) {
            perror(NULL);
            return 0;
        }

        //curl_easy_setopt(curlhandle, CURLOPT_UPLOAD, 1L);

        curl_easy_setopt(curlhandle, CURLOPT_URL, remotepath);

            curl_easy_setopt(curlhandle, CURLOPT_CONNECTTIMEOUT,
timeout);  //  设置连接超时，单位秒
        //设置 http  头部处理函数
        curl_easy_setopt(curlhandle, CURLOPT_HEADERFUNCTION,
getcontentlengthfunc);
        curl_easy_setopt(curlhandle, CURLOPT_HEADERDATA, &filesize);
     //  设置文件续传的位置给 libcurl
        curl_easy_setopt(curlhandle, CURLOPT_RESUME_FROM_LARGE,
use_resume?local_file_len:0);

        curl_easy_setopt(curlhandle, CURLOPT_WRITEDATA, f);
        curl_easy_setopt(curlhandle, CURLOPT_WRITEFUNCTION, wirtefunc);

        //curl_easy_setopt(curlhandle, CURLOPT_READFUNCTION, readfunc);
        //curl_easy_setopt(curlhandle, CURLOPT_READDATA, f);
        curl_easy_setopt(curlhandle, CURLOPT_NOPROGRESS, 1L);
```

```
        curl_easy_setopt(curlhandle, CURLOPT_VERBOSE, 1L);


      r = curl_easy_perform(curlhandle);


        fclose(f);

        if (r == CURLE_OK)
            return 1;
        else {
            fprintf(stderr, "%s\n", curl_easy_strerror(r));
            return 0;
        }
    }

    int main(int c, char **argv) {
        CURL *curlhandle = NULL;

        curl_global_init(CURL_GLOBAL_ALL);
        curlhandle = curl_easy_init();

        //download(curlhandle, "ftp://user:pass@host/path/file", "C:\\file", 0, 3);
      download(curlhandle ,
"http://software.sky-union.cn/index.asp","/work/index.asp",1,3);
        curl_easy_cleanup(curlhandle);
        curl_global_cleanup();

        return 0;
    }
```

编译 **gcc resume.c -o resume －lcurl**

./ resume

3.5LibCurl 调试实例

```
//采用 CURLOPT_DEBUGFUNCTION 参数实现 libcurl 调试功能
#include <stdio.h>
#include <curl/curl.h>

struct data {
  char trace_ascii; /* 1 or 0 */
};
```

```
static
void dump(const char *text,
          FILE *stream, unsigned char *ptr, size_t size,
          char nohex)
{
  size_t i;
  size_t c;

  unsigned int width=0x10;

  if(nohex)
    /* without the hex output, we can fit more on screen */
    width = 0x40;

  fprintf(stream, "%s, %zd bytes (0x%zx)\n", text, size, size);

  for(i=0; i<size; i+= width) {

    fprintf(stream, "%04zx: ", i);

    if(!nohex) {
      /* hex not disabled, show it */
      for(c = 0; c < width; c++)
        if(i+c < size)
          fprintf(stream, "%02x ", ptr[i+c]);
        else
          fputs("   ", stream);
    }

    for(c = 0; (c < width) && (i+c < size); c++) {
      /* check for 0D0A; if found, skip past and start a new line of output */
      if (nohex && (i+c+1 < size) && ptr[i+c]==0x0D && ptr[i+c+1]==0x0A) {
        i+=(c+2-width);
        break;
      }
      fprintf(stream, "%c",
              (ptr[i+c]>=0x20) && (ptr[i+c]<0x80)?ptr[i+c]:'.');
      /* check again for 0D0A, to avoid an extra \n if it's at width */
      if (nohex && (i+c+2 < size) && ptr[i+c+1]==0x0D && ptr[i+c+2]==0x0A)
{
        i+=(c+3-width);
        break;
      }
    }
```

```
    fputc('\n', stream); /* newline */
  }
  fflush(stream);
}

static
int my_trace(CURL *handle, curl_infotype type,
             char *data, size_t size,
             void *userp)
{
  struct data *config = (struct data *)userp;
  const char *text;
  (void)handle; /* prevent compiler warning */

  switch (type) {
  case CURLINFO_TEXT:
    fprintf(stderr, "== Info: %s", data);
  default: /* in case a new one is introduced to shock us */
    return 0;

  case CURLINFO_HEADER_OUT:
    text = "=> Send header";
    break;
  case CURLINFO_DATA_OUT:
    text = "=> Send data";
    break;
  case CURLINFO_SSL_DATA_OUT:
    text = "=> Send SSL data";
    break;
  case CURLINFO_HEADER_IN:
    text = "<= Recv header";
    break;
  case CURLINFO_DATA_IN:
    text = "<= Recv data";
    break;
  case CURLINFO_SSL_DATA_IN:
    text = "<= Recv SSL data";
    break;
  }

  dump(text, stderr, (unsigned char *)data, size, config->trace_ascii);
  return 0;
}
```

```
int main(void)
{
  CURL *curl;
  CURLcode res;
  struct data config;

  config.trace_ascii = 1; /* enable ascii tracing */
```