

followingturing 追随图灵的路上...


悉心求学，博采众长；寡言广学，先博后渊。

目录视图

摘要视图

RSS 订阅

个人资料



followingturing

访问：503260次

积分：6279

等级：BLOG 6

排名：第2471名

原创：68篇

转载：295篇

译文：1篇

评论：70条

文章分类

c# | .net (25)

c/c++ (47)

Java (5)

Linux (54)

WINDWOS编程/MFC/Win32 API (10)

其余未分类的杂~~ (18)

去北邮读研 (10)

实习|简历|面试|找工作 (1)

嵌入式技术 (1)

技术思考 (30)

操作系统 (15)

数据库 (6)

数据结构| 算法|ACM (14)

汇编语言 (17)

硬件描述vhdl及相关 (3)

网络安全与加密破解 (16)

网站设计 (html/css/xml/asp.net) (11)

读病毒 写病毒 (14)

非技术 (4)

项目实习 (2)

飞思卡尔智能车 比赛 (6)

Android (2)

面试\_找工作 (7)

搜索引擎 (10)

python (43)

PHP (22)

文章存档

linux下LibCurl编程

标签：编程 linux gtk stream file thread

2012-08-10 14:17 5133人阅读 评论(0) 收藏 举报

分类： c/c++ (46)

1 LibCurl简介

LibCurl是免费的客户端URL传输库，支持FTP,FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, FILE，LDAP等协议，其主页是<http://curl.haxx.se/>。Libcurl具备线程安全、IpV6兼容、易于使用的特点

介绍LibCurl在http协议方面的应用。

1.1HTTP协议格式

说明：本节主要介绍http协议，若熟悉http协议者可不看。

Http（超文本传输协议）是分布式双向超媒体信息系统应用层协议，主要应用于WWW。通常HTTP消息包括客户机向服务器的请求消息和服务器向客户机的响应消息。http消息（请求或者响应）消息的通用格式实质相同，这两种类型的消息由一个起始行，一个或者多个头域，一个只是头域结束的空行和可选的消息体组成。HTTP的头域包括通用头，请求头，响应头和实体头四个部分。

起始行：请求消息中的的起始行称为请求行，由3个字段组成，它们定义请求的类型、URL和http版本，最后是回车和换行符。请求类型包括get、head、post、put、move等。响应消息中的的起始行称为，也由三个部分组成，http版本、状态码和状态短语，最后是回车和换行符。所有的Http消息题头

http头域：HTTP的头域按其所属性质包括通用头，请求头，响应头和实体头四个部分。通用头域允许出现在请求或者响应消息中，包含Cache-Control、Connection、Date、Pragma、Transfer-Encoding、Upgrade、Via。请求头域只允许出现在请求消息中，响应头域只允许出现在响应消息中，实体头部分提供有消息文档主体信息，主要在响应消息中发送；但是请求消息（如post和put方法）也可以使用实体题头。

每个头域由一个域名，冒号（:）和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。下表为一个典型的请求消息，下面介绍常用头域：

GET http://download.microtool.de:80/somedata.exe HTTP/1.1

Host: download.microtool.de

Accept:/\*/\*

Pragma: no-cache

Cache-Control: no-cache

Referer: http://download.microtool.de/

User-Agent:Mozilla/4.04[en](Win95;I;Nav)

Range:bytes=554554-

消息的第一行是请求行，“GET”表示我们所使用的HTTP动作，其他可能的还有“POST”等，GET的消息没有消息体，而POST消息是有消息体的，消息体的内容就是要POST的数据。后面

http://download.microtool.de:80/somedata.exe就是我们要请求的对象，之后HTTP1.1表示使用的是HTTP1.1协议。

从第2行开始进入http头域，本例中共包括Host、Accept、Pragma、Cache-Control、Referer、User-Agent、Range域。

Host域：表示我们所请求的主机和端口

2012年11月 (11)
2012年10月 (28)
2012年09月 (38)
2012年08月 (23)
2012年07月 (34)
展开

阅读排行
Bootstrap入门教程 (二)
python 之 __str__ (15163)
localhost:8080服务器要: (10194)
北邮电学院[计算机院硕士 (8971)
ubuntu怎么样修改只读文 (8656)
反思"列名无效"的问题 (8400)
Bootstrap入门教程 (三) (7305)
网站安全新隐患——暗链 (6440)
Android和Java ME的区别 (6272)
中软融鑫面试题+笔试题 (5849)

评论排行
我决定工作了 (6)
一个数组越界问题 数组越 (6)
c#打印datagridview, 很 (4)
反思"列名无效"的问题 (3)
教你如何在中关村这个险 (3)
do...while(0)的妙用 (2)
malloc用法 (2)
8086汇编寻址方式教程 (2)
ACM大量习题题库 (2)
人生--机会 (2)

文章搜索
<input type="text"/>

推荐文章
* 浅谈android中异步加载之"取消异步加载"二
*Delta - 轻量级JavaWeb框架使用文档
*Nginx正反向代理、负载均衡等功能实现配置
* 浅析ZeroMQ工作原理及其特点
*android源码解析（十九）-->Dialog加载绘制流程
*Spring Boot 实践折腾记（三）：三板斧，Spring Boot下使用Mybatis

最新评论
linux下替代windows的软件列表 bcbobo21cn: 很多，不错；
白天求生存，晚上谋发展 bluexiaott 学习了
ACM大量习题题库
ironxue: 感觉貌似很有用的东西，顶楼主。
Bootstrap入门教程 (二)
springmvc_springdata: bootstrap demo实例教程源代码下
载:http://www.zuidaima.com/sh...

**Accept**域：表示我们所用的浏览器能接受的Content-type（一般包括image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword），

**Pragma**域：Pragma头域用来包含实现特定的指令，最常用的是Pragma:no-cache。在HTTP/1.1协议中，它的含义和Cache- Control:no-cache相同。

**Cache-Control**域：Cache -Control指定请求和响应遵循的缓存机制。

**Referer**: Referer 头域允许客户端指定请求ui的原资源地址，这可以允许服务器生成回退链表，可用来登陆、优化cache等。他也允许废除的或错误的连接由于维护的目的被追踪。如果请求的uri没有自己的uri地址，Referer不能被发送。如果指定的是部分uri地址，则此地址应该是一个相对地址。

**User-Agent**头域：User-Agent头域的内容包含发出请求的用户信息。

**Range**头域：Range头域可以请求实体的一个或者多个子范围，跟断点续传相关。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

但是服务器可以忽略此请求头，如果无条件GET包含Range，请求头，响应会以状态码206（PartialContent）返回而不是以200（OK）。

一个典型的**响应消息格式**如下：

```
HTTP/1.0200OK
Date:Mon,31Dec200104:25:57GMT
Server:Apache/1.3.14(Unix)
Content-type:text/html
Last-modified:Tue,17Apr200106:46:28GMT
Etag:"a030f020ac7c01:1e9f"
Content-length:39725426
Content-range:bytes554554-40279979/40279980
```

消息第一行为响应行，起格式为如下， 包括http版本、状态码和文本描述。

HTTP-Version SPStatus-CodeSP Reason-Phrase CRLF

HTTP -Version表示支持的HTTP版本，例如为HTTP/1.1。Status- Code是一个三个数字的结果代码。Reason-Phrase给Status-Code提供一个简单的文本描述。Status-Code主要用于机器自动识别，Reason-Phrase主要用于帮助用户理解。Status-Code的第一个数字定义响应的类别，后两个数字没有分类的作用。第一个数字可能取5个不同的值：

1xx:信息响应类，表示接收到请求并且继续处理

2xx:处理成功响应类，表示动作被成功接收、理解和接受

3xx:重定向响应类，为了完成指定的动作，必须接受进一步处理

4xx:客户端错误，客户请求包含语法错误或者是不能正确执行

5xx:服务端错误，服务器不能正确执行一个正确的请求

响应头域允许服务器传递不能放在状态行的附加信息，这些域主要描述服务器的信息和 Request-URI进一步的信息。响应头域包含Age、Location、Proxy-Authenticate、Public、Retry- After、Server、Vary、Warning、WWW-Authenticate。对响应头域的扩展要求通讯双方都支持，如果存在不支持的响应头域，一般将会作为实体头域处理。

在上表中，从第二行开始进入http头域。包括Date、Server、Content-type、Last-modified、Etag、

Content-length、Content-range域。一些比较重要的头域包括：

**Location**域：Location响应头用于重定向接收者到一个新URI地址。

**Content-Type**: 属于实体头，用于向接收方指示实体的介质类型，指定HEAD方法送到接收方的实体介质类型，或GET方法发送的请求介质类型 Content-Range实体头

**Content-length**域：属于实体头，指明实体（文档）长度，表示实际传送的字节数。

Bootstrap入门教程(二)  
sweety820:

在python3下用PIL做图像处理  
wushilonng: 谢谢楼主!很好的资料!

病毒编写教程—1  
zhshuai1: 刚刚读了一小段, 已经忍不住大赞一番了, 语言轻松, 内容详实, 作者的高度让我敬仰。赞!

ILLEGAL\_BP错误详剖析 飞思卡  
柚子2008: 这个问题我也遇到了, 棘手, 但是看到你的解释, 很快解决了, 谢谢分享!

Tesseract限制匹配的字符集  
CCWRSS: 请问下在Windows下可以成功吗?

ACM大量习题题库  
litter\_bird: 今年大二了, 之前没怎么训练过, 现在到大三训练一年, 能有成效吗?

**Content-Range:** 属于实体头, 用于指定整个实体中的一部分的插入位置, 他也指示了整个实体的长度。在服务器向客户返回一个部分响应, 它必须描述响应覆盖的范围和整个实体长度。一般格式:

Content-Range:bytes-unitSP first-byte-pos-last-byte-pos/entity-legth

例如, 传送头500个字节次字段的形式: Content-Range:bytes 0-499/1234如果一个http消息包含此节(例如, 对范围请求的响应或对一系列范围的重叠请求), Content-Range表示传送的范围。

## 2 LibCurl编程

### 2.1 LibCurl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, CURLOPToption 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.

### 3.3 curl\_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。

### 3.4 curl\_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

## 4.实例

### 4.1 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

### 4.2 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`

#### 4.3 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 LibCurl 编程实例

#### 4.4 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头部获取文件大小*/
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 writelfunc(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, writfunc);
}
```

```
//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

#### 4.5 LibCurl 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 */

<font color=&a
```

顶

0

踩

0

- 上一篇
- Linux c 连接处理MYSQL（API方式）
- 下一篇
- python 读取excel表格并写入sqlite数据库

我的同类文章

c/c++（46）

- 一个简单的游戏服务器框架

2012-08-21

阅读 401
- Linux c 连接处理MYSQL ...

2012-08-10

阅读 922
- Linux 抓取网页方式(curl+...

2012-08-09

阅读 628
- Linux下c语言多线程编程

2012-08-01

阅读 1546
- 小议 "undefined reference ...

2012-08-01

阅读 1515
- linux c 多线程执行顺序解析

2012-08-01

阅读 686
- Vim对中文编码的支持

2012-07-28

阅读 375
- SVN遇见的大面积修改的...

2012-07-28

阅读 1750
- gcc提示库里文件未包含 ( ...

2012-07-28

阅读 1399
- BootStrap入门教程 (四)

2012-07-18

阅读 3032

更多文章

猜你在找

- iOS8-Swift开发教程

转载值得推荐的CC++框架和库 真的很强大
- Qt网络编程实战之HTTP服务器

linux下编译libcurl
- Linux高薪入门实战精品 (上)

使用libcurl 编写Linux下Http客户端
- 老郭全套iOS开发课程【Objective-C】

libcurl在windows和linux下的安装
- HTML 5移动开发从入门到精通

Ubuntu1120 32位linux下编译cppunit报

查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker
- OpenStack

VPN

Spark

ERP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC
- WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

HTML

SDK

IIS

Fedora

XML
- LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

Cassandra
- CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App
- SpringSide

Maemo

Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP
- HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django

Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服   杂志客服   微博客服   webmaster@csdn.net   400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持  
京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved 