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

3.5 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 */
```