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
```