WWW Search Engines Part 1: Downloading and Parsing HTML

(for the class Multimedia Information Retrieval by Wu Song)

I. Introduction:

In this tutorial we show the implementation of HTML file download from a given URL address and extract the web-links as URLS from the HTML. The library named curl is used which is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP). In addition, the API of htmlstreamparser is introduced to parse the HTML. For more information (and the authors) about the library of libcurl and htmlstreamparser API, please refer to
http://curl.haxx.se/ and http://code.google.com/p/htmlstreamparser/.

A demo program is given in the C programming language and has been tested to work under Linux and Windows for the machines in room 302. The name of the demo program is htmlprocess.c, and section IV provides the source code of the demo.

II. Important functions:

---

static void Download_URL(char * url, HTMLSTREAMPARSER *hsp)

The job of this function: it first initializes the curl and returns a handle that used as input to other curl_easy functions. Then, it gets the URL address and pass the data from URL to the write_callback function which realizes the process of HTML data. Finally release the all memory allocated previously.

Arguments description:
char * url is a pointer and it points to a char URL[ ]="http://www.liacs.nl/", defined in main function. It passes the URL address to the curl.
HTMLSTREAMPARSER *hsp is pointer and it points to the HTMLSTREAMPARSER structure which defined in the file htmlstreamparser API.

---

static size_t write_callback(char *buffer, size_t size, size_t nmemb, HTMLSTREAMPARSER *hsp)

The job of this function is to extract values of the attribute “href” from the web page, then save the extracted values and print them with standard format when receiving web page data using libcurl function.

Arguments description:
char *buffer: a char pointer and it points to the buffer area which stores the received HTML datas.
size_t size; the size of received data
size_t nmemb, the number of buffer.
HTMLSTREAMPARSER *hsp: a pointer and it points to the HTMLSTREAMPARSER structure.

Descriptions of functions in static size_t write_callback(char *buffer, size_t size, size_t nmemb, HTMLSTREAMPARSER *hsp)

III. Compiling and running the demo program:

If you are using the computers in room 302, then libcurl is already installed. After you unzip the files into a directory, you can

Compile the demo program using:

```
gcc htmlstreamparser.c htmlprocess.c –o htmlprocess –lcurl
```

Run the program with this command:

```
./htmlprocess www.liacs.nl
```

When you run the program, it downloads the contents of www.liacs.nl and prints out the weblinks.

If you want to setup curl on your own computer then here are the instructions. For Linux, you can setup the curl library by:

Step 1: download the library of libcurl (http://curl.haxx.se/download.html)
Please refer to http://curl.haxx.se/docs/install.html to finish the install of libcurl. A normal Linux installation is made in three or four steps (after you've unpacked the source archive); you probably need to be root when doing the last command:

```
./configure
make
make test (optional)
make install
```

You can also do 'make install' without being root. An example of this would be to make a local install in your own home directory:

```
./configure --prefix=$home
make
make install
```

Step 2: download the htmlstreamparser API file from the website as following:
http://code.google.com/p/htmlstreamparser/downloads/detail?name=htmlstreamparser-0.4.tar.gz

Step 3: Once libcurl and htmlstreamparser are installed, you can write your own program combined with the lib and files download from step 1 and 2. Put the files htmlstreamparser.c and htmlstreamparser.h downloaded in step 2 into the folder which contains the file htmlprocess.c. Then, we can run the demo using Linux command line.

Compile the demo program with this command:

```
gcc htmlstreamparser.c htmlprocess.c –o htmlprocess –lcurl
```

Run the program with this command:

```
./htmlprocess www.liacs.nl
```

For windows system, the platform is visual studio 2008
Step 1: add the libcurl into the Visual Studio. How to adding libcurl on Visual Studio 2008, please refer to the pdf document http://curl.haxx.se/libcurl/c/visual_studio.pdf
Step 2: conduct the demo program with the libcurl and the htmlstreamparser API. Create a visual studio project, and then rename the htmlprocess.c and htmlstreamparser.c to htmlprocess.cpp and

htmlstreamparser.cpp. Add the htmlprocess.cpp and htmlstreamparser.cpp as source file into the Visual Studio project; moreover add the htmlstreamparser.h as header file into the project.
Step 3: run the project in Debug mode.

IV. Source code:

htmlprocess.c

```
/* this demo aim:
1. download the html from the URL based on the libcurl;
2. extract the weblinks information from the URL based on the htmlstreamparser API;
In addition, we are planning to add the function extract other types of multimedia informations
*/

/*Header file part */
#include <stdio.h>
#include <string.h>
#include <curl/curl.h>
#include "htmlstreamparser.h"

 FILE *tmp1; /*a pointer to the file save the data of HTML file*/
 FILE *tmp2; /*a pointer to the file save web-links from the URL*/

/*in this part,HTML file saving and web-links extraction are realised. The web-links stored in HTML are indicated by a herf, so we can compare the string data to get the web-links*/
static size_t write_callback(char *buffer, size_t size, size_t nmemb, HTMLSTREAMPARSER *hsp)
{
        /*save the HTML file in tmp1*/
        fwrite(buffer, size, nmemb, (FILE *)tmp1);
        /*the size of the received data*/
        size_t realsize = size * nmemb, p;

        for (p = 0; p < realsize; p++)
        {
                html_parser_char_parse(hsp, ((char *)buffer)[p]);/*Parse the char specified by the char argument*/
                if (html_parser_cmp_tag(hsp, "a", 1)) /*Compares the tag name and the string “a”.The argument 1 is a string length.Returns 1 for equality otherwise returns 0*/
                        if (html_parser_cmp_attr(hsp, "href", 4))/*Compares the attribute name and the string “href”. The argument 4 is a string length.Returns 1 for equality otherwise returns 0*/
                                if (html_parser_is_in(hsp, HTML_VALUE_ENDED))/*Returns 1 if the parser is inside HTML_VALUE_ENDED part of HTML code.*/
                                {
                                        html_parser_val(hsp)[html_parser_val_length(hsp)] = '\0';
                                        printf("%s\n",html_parser_val(hsp));

                                        /*save the weblinks information in tmp2*/

        fwrite(html_parser_val(hsp),html_parser_val_length(hsp),1,(FILE *)tmp2);
                                        fwrite("\n",1,1,(FILE *)tmp2);
                                }
        }
        return realsize;
}
```

```
/*initialization curl and call function to save the HTML file*/
static void Download_URL(char * url,HTMLSTREAMPARSER *hsp)
{
	/*curl_easy_init() initializes curl and this call must have a corresponding call to curl_easy_cleanup();*/
	CURL *curl = curl_easy_init();

	/*tell curl the URL address we are going to download*/
	curl_easy_setopt(curl, CURLOPT_URL, url);

	/*Pass a pointer to the function write_callback( char *ptr, size_t size, size_t nmemb, void *userdata); write_callback gets called by libcurl as soon as there is data received, and we can process the received data, such as saving and weblinks extraction. */
	curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);

   /*it tells the the pointer userdata argument in the write_callback function the data comes from hsp pointer*/
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, hsp);
   /*A parameter set to 1 tells the library to follow any Location: header that the server sends as part of a HTTP header.This means that the library will re-send the same request on the new location and follow new Location: headers all the way until no more such headers are returned.*/
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1);

	/*allow curl to perform the action*/
	CURLcode curl_res = curl_easy_perform(curl);

	if(curl_res==0)
	{
		printf("HTML file downloaded success\n");
	}
	else
	{
		printf("ERROR in dowloading file\n");
		curl_easy_cleanup(curl);
	}
	/*release all the previously allocated memory ,and it corresponds to the function curl_easy_init();*/
	curl_easy_cleanup(curl);
}

/*main function*/
int main(int argc, char *argv[])
{
   /*set the path to save the html file*/
	tmp1=fopen("/tmp/htmldata.html", "w");
	/*set the path to save the extracted web-links*/
	tmp2=fopen("/tmp/weblinksdata.html", "w");
	/*set the URL address to download*/
	//char URL[]="http://www.liacs.nl/~mlew";

	/*a pointer to the HTMLSTREAMPARSER structure and initialization*/
	HTMLSTREAMPARSER *hsp = html_parser_init( );
	char tag[2];
	char attr[5];
	char val[128];
```

```
        html_parser_set_tag_to_lower(hsp, 1);
        html_parser_set_attr_to_lower(hsp, 1);
        html_parser_set_tag_buffer(hsp, tag, sizeof(tag));
        html_parser_set_attr_buffer(hsp, attr, sizeof(attr));
        html_parser_set_val_buffer(hsp, val, sizeof(val)-1);

        /*run download function*/
        Download_URL(argv[1],hsp);

        /*release the hsp*/
        html_parser_cleanup(hsp);

        /*close the file tmp and tmp2 and return*/
        fclose(tmp1);
        fclose(tmp2);
        return EXIT_SUCCESS;
}
```