Bibliotecas para aplicaciones distribuidas

Elaborado por: Ukranio Coronilla

CURL

Para desarrollar aplicaciones distribuidas muchas veces es útil apoyarnos en algunas bibliotecas que se encuentran con desarrollos estables y en muchos casos tienen licencias suficientemente libres para utilizarla en proyectos.

La primera que utilizaremos es la de CURL, cuya página web es la siguiente:

https://curl.haxx.se/

Para instalarlas en UBUNTU solo es necesario ejecutar en la línea de comandos:

```
sudo apt-get install libcurl4-gnutls-dev
```

Básicamente estas bibliotecas tienen implementados muchos protocolos (véase la página web) para ser utilizados en nuestras aplicaciones distribuidas.

En la documentación se encuentra un programa en lenguaje C que hace la función de cliente FTP y se transcribe a continuación.

```
/***************************************************************************
 *                                  _   _ ____  _
 *  Project                     ___| | | |  _ \| |
 *                             / __| | | | |_) | |
 *                            | (__| |_| |  _ <| |___
 *                             \___|\___/|_| \_\_____|
 *
 * Copyright (C) 1998 - 2017, Daniel Stenberg, <daniel@haxx.se>, et al.
 *
 * This software is licensed as described in the file COPYING, which
 * you should have received as part of this distribution. The terms
 * are also available at https://curl.haxx.se/docs/copyright.html.
 *
 * You may opt to use, copy, modify, merge, publish, distribute and/or sell
 * copies of the Software, and permit persons to whom the Software is
 * furnished to do so, under the terms of the COPYING file.
 *
 * This software is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY
 * KIND, either express or implied.
 *
 ***************************************************************************/
#include <stdio.h>

#include <curl/curl.h>

/* <DESC>
 * Get a single file from an FTP server.
 * </DESC>
 */

struct FtpFile {
  const char *filename;
```

```
  FILE *stream;
};

static size_t my_fwrite(void *buffer, size_t size, size_t nmemb, void *stream)
{
  struct FtpFile *out = (struct FtpFile *)stream;
  if(out && !out->stream) {
    /* open file for writing */
    out->stream = fopen(out->filename, "wb");
    if(!out->stream)
      return -1; /* failure, can't open file to write */
  }
  return fwrite(buffer, size, nmemb, out->stream);
}


int main(void)
{
  CURL *curl;
  CURLcode res;
  struct FtpFile ftpfile = {     "walking1.bmp", /* name to store the file as if successful
*/
      NULL
  };

  curl_global_init(CURL_GLOBAL_DEFAULT);

  curl = curl_easy_init();
  if(curl) {
    /*
     * You better replace the URL with one that works!
     */

    curl_easy_setopt(curl, CURLOPT_URL, "ftp://192.168.0.120/walking.bmp");

    curl_easy_setopt(curl, CURLOPT_USERPWD, "escom2017:persefone");
    /* Define our callback to get called when there's data to be written */
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, my_fwrite);
    /* Set a pointer to our struct to pass to the callback */
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, &ftpfile);

    /* Switch on full protocol/debug output */
    curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);

    res = curl_easy_perform(curl);

    /* always cleanup */
    curl_easy_cleanup(curl);

    if(CURLE_OK != res) {
      /* we failed */
      fprintf(stderr, "curl told us %d\n", res);
    }
  }

  if(ftpfile.stream)
    fclose(ftpfile.stream); /* close the local file */

  curl_global_cleanup();

  return 0;
}
```

Observe que en la línea

```
 struct FtpFile ftpfile = {    "walking1.bmp", /* name to store the file as if successful */
```

se especifica como primer miembro de una variable tipo estructura, el nombre que el archivo va a tener al almacenarse en nuestra computadora.

En el tercer parámetro de la línea:

```
curl_easy_setopt(curl, CURLOPT_URL, "ftp://192.168.0.120/walking.bmp");
```

se especifica la dirección de localización uniforme de recursos URL para el esquema FTP.

Finalmente en el tercer parámetro de la línea

```
curl_easy_setopt(curl, CURLOPT_USERPWD, "escom2017:persefone");
```

se especifica el loguin y paswword separado por dos puntos, los cuales son necesarios para acceder al sistema, a menos que esté configurado el usuario anonymous.

Para compilarlo se utiliza la opción de compilación `-lcurl`

Por ejemplo si el programa principal se guarda como programa.c tendríamos que compilar con:

```
gcc programa.c -o programa -lcurl
```

Compílelo y verifique su funcionamiento.

Aquí tenemos otro ejemplo para descargar una página html de mi servidor.

```
#include <stdio.h>
#include <stdlib.h>
#include <curl/curl.h>

int main()
{
        curl_global_init( CURL_GLOBAL_ALL );
        CURL * myHandle;
        CURLcode result; // We'll store the result of CURL's webpage retrieval, for simple
error checking.
        myHandle = curl_easy_init ( ) ;
        // Notice the lack of major error checking, for brevity
        curl_easy_setopt(myHandle, CURLOPT_URL, "http://192.168.0.120/incondicional.pdf");
        result = curl_easy_perform( myHandle );
        curl_easy_cleanup( myHandle );
        printf("LibCurl rules!\n");
        return 0;
}
```

Observe en la línea siguiente que el tercer parámetro es la dirección URL en el esquema HTML.

```
curl_easy_setopt(myHandle, CURLOPT_URL, "http://192.168.0.120/incondicional.pdf");
```

Dado que el protocolo HTML normalmente permite obtener información visible por el navegador y se está intentando traer un archivo pdf, es necesario almacenar la información recibida en un archivo. Para eso utilizamos el operador redirección de UNIX como sigue:

```
./programa > incondicional.pdf
```

MONGOOSE

Muchas veces es necesario tener nuestro propio servidor HTML incrustado en nuestro código para poder brindar una interfaz web amigable a los clientes. Para ello utilizaremos el código Mongoose disponible en:

https://cesanta.com/docs/overview/intro.html

Solo descargue los archivos mongoose.c y mongoose.h cuyas ligas se encuentran en la página principal y se transcriben a continuación:

https://raw.githubusercontent.com/cesanta/mongoose/master/mongoose.c

https://raw.githubusercontent.com/cesanta/mongoose/master/mongoose.h

Podrías guardar estos códigos con el programa anterior… :-D

Estos códigos se compilan y se ligan con nuestro código fuente, de los cuales hay muchos ejemplos en el sitio web. El más simple es el siguiente:

```
// Copyright (c) 2015 Cesanta Software Limited
// All rights reserved

#include "mongoose.h"

static const char *s_http_port = "8000";
static struct mg_serve_http_opts s_http_server_opts;

static void ev_handler(struct mg_connection *nc, int ev, void *p) {
  if (ev == MG_EV_HTTP_REQUEST) {
    mg_serve_http(nc, (struct http_message *) p, s_http_server_opts);
  }
}

int main(void) {
  struct mg_mgr mgr;
  struct mg_connection *nc;

  mg_mgr_init(&mgr, NULL);
  printf("Starting web server on port %s\n", s_http_port);
  nc = mg_bind(&mgr, s_http_port, ev_handler);
  if (nc == NULL) {
    printf("Failed to create listener\n");
    return 1;
  }

  // Set up HTTP server parameters
  mg_set_protocol_http_websocket(nc);
  s_http_server_opts.document_root = ".";  // Serve current directory
  s_http_server_opts.enable_directory_listing = "yes";

  for (;;) {
    mg_mgr_poll(&mgr, 1000);
  }
  mg_mgr_free(&mgr);

  return 0;
}
```

Observe que en la línea

```
static const char *s_http_port = "8000";
```

estoy especificando el puerto del servidor HTTP como el 8000. Para compilar ejecuto:

gcc mongoose.c servidorHTML.c -o servidorHTML

y listo, al ejecutarlo tengo un servidor HTML en el puerto 8000; lo cual puedo comprobar poniendo en mi navegador la siguiente dirección:

http://127.0.0.1:8000/

En la página web vienen más ejemplos interesantes como el que está disponible en mi servidor y cuyo nombre es:

mongoose_ejemplo.tar

Pruebelo ¡¡¡¡¡