multithread.c

Download multithread.c raw

```
/***************************************************************************
 *                                  _   _ ____  _
 *  Project                     ___| | | |  _ \| |
 *                             / __| | | | |_) | |
 *                            | (__| |_| |  _ <| |___
 *                             \___|\___/|_| \_\_____|
 *
 * Copyright (C) 1998 - 2015, 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.
 *
 ***************************************************************************/
/* <DESC>
 * A multi-threaded example that uses pthreads to fetch several files at once
 * </DESC>
 */

#include <stdio.h>
#include <pthread.h>
#include <curl/curl.h>

#define NUMT 4

/*
  List of URLs to fetch.

  If you intend to use a SSL-based protocol here you MUST setup the OpenSSL
  callback functions as described here:

  https://www.openssl.org/docs/crypto/threads.html#DESCRIPTION

*/
const char * const urls[NUMT]= {
  "https://curl.haxx.se/",
  "ftp://cool.haxx.se/",
  "http://www.contactor.se/",
  "www.haxx.se"
};

static void *pull_one_url(void *url)
{
  CURL *curl;

  curl = curl_easy_init();
  curl_easy_setopt(curl, CURLOPT_URL, url);
  curl_easy_perform(curl); /* ignores error */
  curl_easy_cleanup(curl);

  return NULL;
}


/*
   int pthread_create(pthread_t *new_thread_ID,
   const pthread_attr_t *attr,
   void * (*start_func)(void *), void *arg);
*/

int main(int argc, char **argv)
{
  pthread_t tid[NUMT];
  int i;
  int error;

  /* Must initialize libcurl before any threads are started */
  curl_global_init(CURL_GLOBAL_ALL);

  for(i=0; i< NUMT; i++) {
    error = pthread_create(&tid[i],
                           NULL, /* default attributes please */
                           pull_one_url,
                           (void *)urls[i]);
    if(0 != error)
      fprintf(stderr, "Couldn't run thread number %d, errno %d\n", i, error);
    else
      fprintf(stderr, "Thread %d, gets %s\n", i, urls[i]);
  }

  /* now wait for all threads to terminate */
  for(i=0; i< NUMT; i++) {
    error = pthread_join(tid[i], NULL);
    fprintf(stderr, "Thread %d terminated\n", i);
  }

  return 0;
}
```