multithreaded/curl_xml.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <curl/curl.h>
#include <libxml/HTMLparser.h>
#include <libxml/parser.h>
#include <libxml/xpath.h>
#include <libxml/uri.h>
#include "stack.h"

#define SEED_URL "http://ece252-1.uwaterloo.ca/lab4/"
#define ECE252_HEADER "X-Ece252-Fragment: "
#define CURL_USER_AGENT_FIELD "ece252 lab4 crawler"
#define BUF_SIZE 1048576 /* 1024*1024 = 1M */
#define BUF_INC 524288   /* 1024*512  = 0.5M */

#define CT_PNG "image/png"
#define CT_HTML "text/html"
#define CT_PNG_LEN 9
#define CT_HTML_LEN 9
#define URL_LENGTH 256

#define OK_REQUESTS 200
#define REDIRECT_REQUESTS 300
#define BAD_REQUESTS 400
#define INTERNAL_SERVER_ERRORS 500
#define CODE_RANGE 99

#define DEFAULT_TYPE -1
#define HTML 0
#define VALID_PNG 1
#define INVALID_PNG 2

#define max(a, b) \
  ({ __typeof__ (a) _a = (a); \
      __typeof__ (b) _b = (b); \
    _a > _b ? _a : _b; })

typedef struct recv_buf2
{
  char *buf;       // memory to hold a copy of received data
  size_t size;     // size of valid data in buf in bytes
  size_t max_size; // max capacity of buf in bytes
  int seq;         // >=0 sequence number extracted from http header
                   // <0 indicates an invalid seq number
```

```
49 } RECV_BUF;
50
51 htmlDocPtr mem_getdoc(char *buf, int size, const char *url);
52 xmlXPathObjectPtr getnodeset(xmlDocPtr doc, xmlChar *xpath);
53 int find_http(char *fname, int size, int follow_relative_links, const char *base_url, STACK
   *stack);
54 size_t header_cb_curl(char *p_recv, size_t size, size_t nmemb, void *userdata);
55 size_t write_cb_curl(char *p_recv, size_t size, size_t nmemb, void *p_userdata);
56 int recv_buf_init(RECV_BUF *ptr, size_t max_size);
57 int recv_buf_cleanup(RECV_BUF *ptr);
58 void cleanup(CURL *curl, RECV_BUF *ptr);
59 CURL *easy_handle_config(CURL *curl_handle, RECV_BUF *ptr, const char *url);
60 int process_data(CURL *curl_handle, RECV_BUF *p_recv_buf, int *content_type, STACK *stack, long
   *response_code_p);
61 int process_png(CURL *curl_handle, RECV_BUF *p_recv_buf, int *content_type);
62 bool is_png(uint8_t *buf, size_t n);
63 int process_url(CURL *curl_handle, char *seed_url, int *content_type, STACK *stack, long
   *response_code_p);
64 bool is_processable_response(long response_code);
```