```c
#ifndef _PARSER_H_
#define _PARSER_H_

#define BUFLEN               (256)      /* buffer length */
#define TIME_BUFFER_LEN     128     /* lenght of time buffer to print time stamp*/

typedef struct {
    int status;
    size_t length;
    char *type;
    char *host;
} header;

enum commands {
    INVALID,
    LIST,
    GET,
    COMMENT,
    EXIT
};

typedef enum commands command_t;

/*
 * A contructor function for header struct
 * @returns an empty header struct
 */
header create_header();

/*
 * @param request - string line to validate
 * @returns
 *      1 if valid, -1 if not
 */
command_t get_command_from_request(const char *request);

/*
 * @param header - buffer containing header information
 * @param line_number - spefic line of header buffer to return
 * @returns
 *      a particular line from header buffer
 */
char * get_line(char * header_text, unsigned int line_number);

/*
 * @param string - buffer to find occurence of chracter from
 * @param c      - value of char whose occurence to be found
 * @param n      - number of occurences to be found
 * @returns      - position index of the nth occurence.
 */
int get_occurrence_n(char * string, char c, int n);

/*
 *  @param buf - buffer to store the time spec in
 *  @param buflen - size of the buffer
 */
void get_time_spec_to_string(char *buf, size_t buflen);

/*
 * @param str - string to find the number of lines it contains
 * @returns   - number of lines in a string
 */
int count_lines(char const *str);

/*
 * @param socket - socket id to receive header text from
 * @returns      - prints and then returns a buffer containing header text
 */
char * read_header_text(int socket);
```

```c

/*
 * @param header_text - buffer to read from
 * @param header_ptr  - storage location to store information
 * @returns           - success or failure
 */
int buffer_to_header(char * header_text, header *ptr);

/* Handle command from a string value
 * @param socker        - socket to use for server communication
 * @param command       - command string read from usr or file
 * @param len           - len of incoming command
 * @returns             - success or failure
 */
int handle_command(int socket, char *command, int len);

/*
 * Handle any command request from client
 * @param server_socket - socket to communicate to server
 * @param command       - string containing the full get <filename>
 * @param               - strlen of command
 * @returns - success or failure
 */
int process_command(int server_socket, char *command, int len);

/*
 * Runs commands from batch script
 * @param clientrc_path - path to read client commands from
 */
int process_batch(int socket, char * clienrc_path);

#endif
```