Lab 6

Distint Howie
Robert Krency
Anthony Stepich

Installation

To install, create a folder to copy files into. Clone the repository into the directory with the following command:
`git clone https://github.com/rdk-fall22-calu/cet440-server-phase3.git`
To compile, run the `make` command inside of the `build` directory.
To run, launch the server program inside the `build` directory: `.\server`

Users File

The server program requires a list of users to be present in the `build` directory, titled `names.txt` which contains usernames and full names. The provided file can be found at this link
(`https://students.calu.edu/calupa/chen/cet440/lab/studentlist.txt`),
and contains the following information:

```
chen Weifeng Chen
bar4167 Patrick Barker
bat6731 Luke Bates
bis3683 Andrew Bissell
bra8956 Scott Bracker
cas3742 Branson Casper
gro6315 Ryan Groner
gue4713 David Guest
gun4897 Justin Gunderson
how4685 Distint Howie
hug8634 Joshua Hughes
hut2029 Jack Hutchinson
kir0510 Tanner Kirsch
kov2428 Camden Kovach
kre1188 Robert Krency
kre5277 Ty Kress
kru2922 Charles Krug
min5779 Madeline Minsinger
nie9236 Noelle Nieves
pri2679 Austin Pringle
pro8061 Noah Proctor
rei3819 Kevin Reisch
sea3212 Kitana Seals
ste4864 Anthony Stepich
tei3216 Zachary Teixido
ter1023 Thomas Terhune
tru1931 Scott Trunzo
```

Cipher Key

The program also requires the following cipher key to encrypt and decrypt all communications.

```
    78 4 3 73 1 28 81 32 21 74 26 23 94 55 40 9 35 47 5 39 16 8 50 91 66 58 75 79 41 63 15 43 69
    68 42 46 33 59 22 6 54 52 19 56 7 77 85 44 20 67 86 14 93 24 62 83 30 12 90 37 84 76 72 25 10
    18 89 87 71 49 92 57 88 60 65 31 27 64 17 36 45 61 82 11 29 38 53 34 2 13 51 0 80 48 70
```

Header Files

commands.h

A list of accepted commands can be found at: `https://students.calu.edu/calupa/chen/cet440/lab/protocol.pdf`

```
/**
* @brief Returns a string with all of the help information
*
* @return char* Response string
*/
char* execute_help();

/**
* @brief If the user is logged into the server, it logs them out and returns a disconnecting message.
*
* @return char* Response string
*/
char* execute_quit();

/**
* @brief The server will attempt to register them and respond with a string telling them that their userID has been registered successfully or not possible reasons are it was already registered or it is an invalid userID.
*
* @param userID
* @return char* Response string
*/
char* execute_register(char* userID, int socket);

/**
* @brief The server will attempt to unregister the user and set its data back to defaults.
*
* @param userID
* @return char* Response string
*/
char* execute_unregister(char* userID);

/**
* @brief The server will respond with a string telling thme if they had a successful login. Requires registered user.
*
* @param userID
* @return char* Response string
*/

char* execute_login(char*userID, int socket);

/**
* @brief The server will respond with a string of information about the user. Requires registered user.
*
* @param userID
* @return char* Response string
*/

char* execute_myinfo(char* userID);

/**
* @brief The server will respond with a string of all online users. Requires registered user.
*
* @param userID
* @return char* Response string
*/
char* execute_online_users(char* userID);

/**
* @brief The server will respond with a string of all registered users. Requires registered user.
*
* @param userID
```

```
* @return char* Response string
*/
char* execute_registered_users(char* userID);
```

logging.h

```
/**
 * @brief Log the formatted message to the console
 *
 * @param sender The sender's ID
 * @param message Message to be logged
 */
void log_message(char *sender, char *message);
```

users.h

```
/**
 * @brief Defines a User
 *
 */
struct user {
    char userID[10];
    char name[50];
    int age;
    float gpa;
    char address[50]; // IP Address user is connected from
    int status;
    char password[PWD_SIZE];
};

static struct user userList[NUM_USERS];

/**
 * @brief Loads the list of accepted users from the file names.txt
 *
 * @return int 1 for successful loading, 0 for failed to load
 */
int load_users_list();


/**
 * @brief Initializes random values for each user
 */
int initialize_users();


/**
 * @brief Saves the user data to a CSV file.
 *
 */
void save_user_data();


/**
 * @brief Returns the user with the specified user ID
 *
 * @param userID
 * @return Pointer to user struct, NULL if user does not exist
 */
struct user* get_user(char *userID);


/**
 * @brief Gets the list of users
 *
 * @return user* pointer to the user list
 */
struct user* get_user_list();
```

server.h

```c
/**
* @brief Contains the IP Address and Socket for the specific client
*
*/
struct clientInfo {
    char ipaddress[15];
    void *socket;
};

/**
* @brief Set up server connection
*
*/
int setup_server();


/**
* @brief Starts the server
*
* @param connection_handler The function to call for connection handling
*/
int start_server(void * (*connection_handler)(void *));


/**
* @brief Encrypts the message to be sent, and sends it to the specified socket
*
* @param socket
* @param message
*
*/
void send_message(int socket, char *message);


/**
* @brief Receive a message from the client
*
* @param socket
* @param message
* @return int Size of the message received
*/
int receive_message(int socket, char *message);
```

main.c

```c
/**
* @brief Threading function
*
* @return void*
*/
void *connection_handler(void *);

/**
* @brief Convert a string to all upper case
*
* @param text
* @return char*
*/
char *strupr(char * text);

/**
* @brief Entry point and main loop for the server
*
* @return int Exit message
*/
int main();
```

cipher.h

```
struct cipher {
    char encode_list[95];
    char decode_list[95];
};

/**
 * @brief Load the cipher key from memory and initialize the cipher
 *
 */
int load_cipher();


/**
 * @brief Encode the text into enc using the substitution cipher
 *
 * @param text
 */
void encode(char* text, char* enc);


/**
 * @brief Decode the text into dec using the substitution cipher
 *
 * @param text
 */
void decode(char* text, char* dec);
```

Contributions

Distint Howie	Testing
Robert Krency	Encryption, Passwords, Commands, Testing, Accompanying Documentation
Anthony Stepich	Commands, Testing