

Net-Centric Lab 3

Student: Nguyen Duc Toan

ID: ITCSIU21112

Client.go

```
1  package main
2
3  import (
4      "fmt"
5      "io"
6      "net"
7      "os"
8      "strings"
9  )
10
11  const (
12      HOST = "localhost"
13      PORT = "8080"
14      TYPE = "tcp"
15  )
16
17  func main() {
18      tcpServer, err := net.ResolveTCPAddr(TYPE, HOST+":"+PORT)
19
20      if err != nil {
21          println("ResolveTCPAddr failed:", err.Error())
22          os.Exit(1)
23      }
24
25      conn, err := net.DialTCP(TYPE, nil, tcpServer)
26      if err != nil {
27          println("Dial failed:", err.Error())
28          os.Exit(1)
29      }
30
31      defer conn.Close() // Ensure connection is closed when the main function returns
32
33      // Login or register
34      fmt.Println("1. Login")
35      fmt.Println("2. Register")
36      fmt.Println("3. Exit")
37      fmt.Print("Enter your choice: ")
38      var choice int
39      fmt.Scanln(&choice)
40
41      switch choice {
42      case 1:
43          username, password := login()
44
45          // Send the login data to the server
46          _, err = conn.Write([]byte("Login" + "|" + username + "|" + password))
47          if err != nil {
48              println("Write data failed:", err.Error())
49              os.Exit(1)
50          }
51      }
```

```

52     received := make([]byte, 1024)
53     _, err = conn.Read(received)
54     if err != nil {
55         println("Read data failed:", err.Error())
56         os.Exit(1)
57     }
58     fmt.Println(string(received))
59
60     if strings.Contains(string(received), "successful") {
61         fmt.Println("Hello client from server")
62         fmt.Println("1. Play Game")
63         fmt.Println("2. Download File")
64         fmt.Println("3. Exit")
65
66         fmt.Print("Enter your choice: ")
67         var choice int
68         fmt.Scanln(&choice)
69
70         switch choice {
71         case 1:
72             _, err = conn.Write([]byte("Game"))
73             fmt.Println("Welcome to Guessing Game!")
74             playGame(err, conn)
75
76         case 2:
77             _, err = conn.Write([]byte("Download"))
78             fmt.Print("Enter the name of the file to download: ")
79             var fileName string
80             fmt.Scanln(&fileName)
81
82             // Send download request
83             _, err = conn.Write([]byte(fileName))
84             if err != nil {
85                 fmt.Println("Error sending file request:", err)
86                 return
87             }
88
89             // Receive the file data
90             requestFile(conn, fileName)
91         case 3:
92             _, err = conn.Write([]byte("Exit"))
93             exit()
94         default:
95             println("Invalid choice")
96             os.Exit(0)
97         }
98     }

```

```

100     case 2:
101         username, password, fullname, email, address := register()
102
103         // Send the register data to the server
104         _, err = conn.Write([]byte("Register" + "|" + username + "|" + password + "|" + fullname + "|" + email + "|" + address))
105         if err != nil {
106             println("Write data failed:", err.Error())
107             os.Exit(1)
108         }
109     case 3:
110         exit()
111     default:
112         println("Invalid choice")
113         return
114     }
115
116     conn.Close()
117 }
118 }
119
120 func login() (username, password string) {
121     // Login to server
122     fmt.Print("Enter username: ")
123     // var username string
124     fmt.Scanln(&username)
125
126     fmt.Print("Enter password: ")
127     // var password string
128     fmt.Scanln(&password)
129
130     return username, password
131 }
132
133 func register() (username, password, fullname, email, address string) {
134     // Register to server
135     fmt.Print("Enter username: ")
136     fmt.Scanln(&username)
137
138     fmt.Print("Enter password: ")
139     fmt.Scanln(&password)
140
141     fmt.Print("Enter fullname: ")
142     fmt.Scanln(&fullname)
143
144     fmt.Print("Enter email: ")
145     fmt.Scanln(&email)
146
147     fmt.Print("Enter address: ")
148     fmt.Scanln(&address)

```

```

149
150     return username, password, fullname, email, address
151 }
152
153 func exit() {
154     return
155 }
156
157 func playGame(err error, conn *net.TCPConn) {
158     received := make([]byte, 1024)
159     _, err = conn.Read(received)
160     for {
161         fmt.Printf("Guess a number between 1 and 100 (press 0 to exit): ")
162         var guessNumber int
163         fmt.Scanln(&guessNumber)
164
165         if guessNumber == 0 {
166             _, err = conn.Write([]byte("Exit"))
167             return
168         }
169
170         _, err = conn.Write([]byte(fmt.Sprintf("%d", guessNumber)))
171         if err != nil {
172             println("Write data failed:", err.Error())
173             os.Exit(1)
174         }
175
176         // Read the response from the server
177         received := make([]byte, 1024)
178         _, err = conn.Read(received)
179         if err != nil {
180             println("Read data failed:", err.Error())
181             os.Exit(1)
182         }
183
184         response := strings.TrimSpace(string(received[:]))
185         fmt.Printf("Received response: '%s'\n", response)

```

```

186
187         if strings.Contains(response, "_Congratulations!") {
188
189             fmt.Println("Do you want to play again? (y/n)")
190             var playAgain string
191             fmt.Scanln(&playAgain)
192
193             if strings.ToLower(playAgain) == "n" {
194                 _, err = conn.Write([]byte("Exit")) // Send "Exit" to the server
195                 break
196             } else if strings.ToLower(playAgain) == "y" {
197                 // If the user says yes, send "Again" to the server
198                 _, err = conn.Write([]byte("Again"))
199                 _, err = conn.Read(received) // Wait for server's response to start the game again
200                 fmt.Println("Starting a new game...")
201                 conn.Read(received)
202             }
203         }
204     }
205 }
206

```

```

207 func requestFile(conn net.Conn, fileName string) {
208     // Create or open the file for writing (only if the file exists)
209     outFile, err := os.Create("downloaded_" + fileName)
210     if err != nil {
211         fmt.Println("Error creating file:", err)
212         return
213     }
214     defer outFile.Close()
215
216     // Buffer to receive the file data
217     buffer := make([]byte, 1024)
218
219     // Read the response from the server
220     for {
221         n, err := conn.Read(buffer)
222         // Handle server responses
223         serverResponse := string(buffer[:n])
224         // fmt.Print("Server response: ", serverResponse)
225         if err != nil {
226             if err == io.EOF {
227                 fmt.Println("File download completed.")
228                 break
229             }
230             fmt.Println("Error reading from server:", err)
231             return
232         }
233
234         // Ignore notification messages from the server
235         if serverResponse == "File download starting...\n" || serverResponse == "\nFile download complete\n" {
236             continue
237         }
238         if serverResponse == "Error: File not found\n" {
239             fmt.Println("Server response: ", serverResponse)
240             return // Exit if the file does not exist
241         }
242
243         // Write file data to the local file if it's not an error message
244         outFile.Write(buffer[:n])
245     }
246 }
247

```

Server.go

```
1 package main
2
3 import (
4     "encoding/base64"
5     "encoding/json"
6     "fmt"
7     "io"
8     "log"
9     "math/rand"
10    "net"
11    "os"
12    "strconv"
13    "strings"
14 )
15
16 const (
17     HOST = "localhost"
18     PORT = "8080"
19     TYPE = "tcp"
20 )
21
22 type User struct {
23     Username string `json:"username"`
24     Password string `json:"password"`
25     Fullname string `json:"fullname"`
26     Email    []string `json:"email"`
27     Address  []string `json:"address"`
28 }
29
30 var randomKey = rand.Intn(1000) + 1
31 var key = strconv.Itoa(randomKey)
32
33 func main() {
34     listen, err := net.Listen(TYPE, HOST+":"+PORT)
35     if err != nil {
36         log.Fatal(err)
37         os.Exit(1)
38     }
39     // close listener
40     defer listen.Close()
41     for {
42         conn, err := listen.Accept()
43         if err != nil {
44             log.Fatal(err)
45             os.Exit(1)
46         }
47
48         go handleRequest(conn)
49     }
50 }
51
```

```

52 func handleRequest(conn net.Conn) {
53     defer conn.Close()
54
55     // incoming request
56     buffer := make([]byte, 1024)
57     _, err := conn.Read(buffer)
58     if err != nil {
59         log.Fatal(err)
60     }
61
62     clientMsg := string(buffer)
63
64     if clientMsg[:5] == "Login" {
65         username, password := handleLogin(clientMsg)
66
67         if authenticateUser(username, password) {
68             fmt.Println(key + "_Hello Server from client")
69             _, err = conn.Write([]byte(key + "_Authentication successful\n"))
70             conn.Read(buffer) // Read the next message from the client
71             clientMsg := strings.TrimSpace(string(buffer))
72             if strings.HasPrefix(clientMsg, "Game") {
73                 startGuessingGame(err, conn)
74             } else if strings.HasPrefix(clientMsg, "Download") {
75                 startFileTransfer(conn)
76             } else if strings.HasPrefix(clientMsg, "Exit") {
77                 return
78             }
79         } else {
80             fmt.Println(key + "_Login failed")
81             _, err = conn.Write([]byte(key + "_Login failed"))
82         }
83
84     } else if strings.HasPrefix(clientMsg, "Register") {
85         handleRegister(clientMsg)
86         conn.Write([]byte("_Register successful"))
87     }
88 }
89
90 func encryptPassword(password string) string {
91     return base64.StdEncoding.EncodeToString([]byte(password))
92 }
93
94 func decryptPassword(encryptedPassword string) string {
95     decoded, _ := base64.StdEncoding.DecodeString(encryptedPassword)
96     return string(decoded)
97 }
98

```

```

99 func loadUsers(filename string) []User {
100     jsonFile, err := os.Open(filename)
101
102     // Check if the file exists
103     if err != nil {
104         log.Fatal(err)
105     }
106
107     data, _ := io.ReadAll(jsonFile)
108     var users []User
109     json.Unmarshal(data, &users)
110     return users
111 }
112
113 func authenticateUser(username, password string) bool {
114     users := loadUsers("User.json")
115     encryptedPassword := encryptPassword(password)
116     for _, user := range users {
117         if user.Username == username && user.Password == encryptedPassword {
118             return true
119         }
120     }
121     return false
122 }
123
124 func handleLogin(clientMsg string) (username, password string) {
125     data := strings.Split(clientMsg, "|")
126     username = strings.TrimSpace(data[1])
127     password = strings.Trim(data[2], "\x00")
128
129     return username, password
130 }
131
132 func saveUser(user User, filename string) {
133     users := loadUsers(filename)
134     users = append(users, user)
135     data, _ := json.Marshal(users)
136     _ = os.WriteFile(filename, data, 0644)
137 }
138

```

```

139 func handleRegister(clientMsg string) {
140     data := strings.Split(clientMsg, "|")
141     username := strings.TrimSpace(data[1])
142     password := strings.Trim(data[2], "\x00")
143     fullname := strings.TrimSpace(data[3])
144     email := strings.Split(strings.TrimSpace(data[4]), ",")
145     address := strings.Split(strings.Trim(data[5], "\x00"), ",")
146
147     user := User{
148         Username: username,
149         Password: encryptPassword(password),
150         Fullname: fullname,
151         Email:    email,
152         Address:  address,
153     }
154
155     saveUser(user, "User.json")
156 }

```



```

158 func startGuessingGame(err error, conn net.Conn) {
159     for {
160         // Send the message to the client, to start the game
161         conn.Write([]byte("Start"))
162
163         // Generate random number
164         randomNumber := rand.Intn(100) + 1
165         target := randomNumber
166         fmt.Println("Target number:", target)
167         for {
168             received := make([]byte, 1024)
169             _, err = conn.Read(received)
170             if err != nil {
171                 log.Fatal(err)
172             }
173
174             clientMsg := strings.TrimSpace(string(received[:]))
175
176             if strings.Contains(clientMsg, "Exit") {
177                 break
178             }
179
180             if strings.Contains(clientMsg, "Again") {
181                 break
182             }
183
184             // Convert received data to integer
185             // guessNumber, err := strconv.Atoi(string(received))
186             guessNumber := 0
187             fmt.Sscanf(string(received), "%d", &guessNumber)
188             fmt.Println("Guess number:", guessNumber)
189
190             if guessNumber < target {
191                 conn.Write([]byte(key + "_Your guess is too low. Try again. "))
192             } else if guessNumber > target {
193                 conn.Write([]byte(key + "_Your guess is too high. Try again. "))
194             } else {
195                 conn.Write([]byte(key + "_Congratulations! You guessed the number. "))
196                 break
197             }
198         }
199     }
200 }
201 }
202 }
203

```

```

func startFileTransfer(conn net.Conn) {
    // Read the requested file name from the client
    buffer := make([]byte, 1024)
    n, err := conn.Read(buffer)
    if err != nil {
        fmt.Println(key+"_Error reading from client:", err)
        return
    }
    fileName := string(buffer[:n])
    fmt.Println("Client requested file:", fileName)

    // Check if the file exists on the server
    if _, err := os.Stat(fileName); os.IsNotExist(err) {
        // File does not exist, notify the client
        conn.Write([]byte("Error: File not found\n"))
        fmt.Println(key+"_File not found:", fileName)
        return
    }

    // File exists, start the file transfer
    file, err := os.Open(fileName)
    if err != nil {
        fmt.Println("Error opening file:", err)
        conn.Write([]byte("Error: Unable to open file\n"))
        return
    }
    defer file.Close()

    // Notify client that file is starting to download
    conn.Write([]byte("File download starting...\n"))

    // Send file data in chunks
    buffer = make([]byte, 1024)
    for {
        n, err := file.Read(buffer)
        if err != nil && err != io.EOF {
            fmt.Println("Error reading file:", err)
            break
        }
        if n == 0 {
            break
        }
        conn.Write(buffer[:n]) // Send the context to the client
    }

    // Notify client the file transfer is complete
    conn.Write([]byte("\nFile download complete\n"))
    fmt.Println("File download completed for:", fileName)
}

```

User.json

```
1  [
2      {
3          "username": "admin",
4          "password": "YWRtaW4=",
5          "fullname": "Administrator",
6          "email": [
7              "admin@email.com",
8              "ad@email.com"
9          ],
10         "address": [
11             "HCM"
12         ]
13     }
14 ]
```

Result:

```
PS E:\IU\Senior\Wet-Centric Lab\Lab3> cd .\client\
PS E:\IU\Senior\Wet-Centric Lab\Lab3\client> go run .\client.go
1. Login
2. Register
3. Exit
Enter your choice: 1
Enter username: admin
Enter password: admin
477_Authentication successful

Hello client from server
1. Play Game
2. Download File
3. Exit
Enter your choice: 1

PS E:\IU\Senior\Wet-Centric Lab\Lab3\server>
* History restored

PS E:\IU\Senior\Wet-Centric Lab\Lab3>
* History restored

PS E:\IU\Senior\Wet-Centric Lab\Lab3> cd .\server\
PS E:\IU\Senior\Wet-Centric Lab\Lab3\server> go run .\server.go
477_Hello Server from client
```

Guessing Game:

```
Hello client from server
1. Play Game
2. Download File
3. Exit
Enter your choice: 1
Welcome to Guessing Game!
Guess a number between 1 and 100 (press 0 to exit): 20
Received response: '477_Your guess is too low. Try again.'
Guess a number between 1 and 100 (press 0 to exit): 30
Received response: '477_Your guess is too high. Try again.'
Guess a number between 1 and 100 (press 0 to exit): 25
Received response: '477_Congratulations! You guessed the number.'
Do you want to play again? (y/n)
y

PS E:\IU\Senior\Wet-Centric Lab\Lab3>
* History restored

PS E:\IU\Senior\Wet-Centric Lab\Lab3>
* History restored

PS E:\IU\Senior\Wet-Centric Lab\Lab3> cd .\server\
PS E:\IU\Senior\Wet-Centric Lab\Lab3\server> go run .\server.go
477_Hello Server from client
Target number: 25
Guess number: 20
Guess number: 30
Guess number: 25
Target number: 54
```

```
1. Play Game
2. Download File
3. Exit
Enter your choice: 1
Welcome to Guessing Game!
Guess a number between 1 and 100 (press 0 to exit): 93
Received response: '661_Congratulations! You guessed the number.'
Do you want to play again? (y/n)
y
Starting a new game...
Guess a number between 1 and 100 (press 0 to exit): 98
Received response: '661_Congratulations! You guessed the number.'
Do you want to play again? (y/n)
n
PS E:\IU\Senior\Wet-Centric Lab\Lab3\client>
```

File download:

```
2. Register
3. Exit
Enter your choice: 1
Enter username: admin
Enter password: admin
200_Authentication successful

Hello client from server
1. Play Game
2. Download File
3. Exit
Enter your choice: 2
Enter the name of the file to download: text.txt
File download completed.
PS E:\IU\Senior\Net-Centric Lab\Lab3\client>
```

```
PS E:\IU\Senior\Net-Centric Lab\Lab3\server> go run .\server.go
200_Hello Server from client
Client requested file: text.txt
File download completed for: text.txt
```

- Request File:

```
server.go x text.txt x
server > text.txt
1 ITCIU21112
2 Nguyen Duc Toan
3 Net-centric Programming
```

- Downloaded File:

```
client.go U downloaded_text.txt U x
client > downloaded_text.txt
1 ITCIU21112
2 Nguyen Duc Toan
3 Net-centric Programming
```