Netcentric lab 6

Nguyen Manh Viet Khoi
ITCSIU21081

```go
// Package main provides a simple HTTP server.
package main

import (
    "bufio"
    "fmt"
    "net"
    "os"
    "strconv"
    "strings"
)

// main starts the server and listens for connections on
TCP port 9999.
func main() {
    // Listen on TCP port 9999
    ln, err := net.Listen("tcp", ":9999")
    if err != nil {
        fmt.Println(err)
        return
    }
    defer ln.Close()
    fmt.Println("Server is listening on port 9999")

    // Accept connections in a loop and handle each one in
a separate goroutine.
    for {
        conn, err := ln.Accept()
        if err != nil {
            fmt.Println(err)
            return
        }
        go handleConnection(conn)
    }
}

// handleConnection handles an individual client
connection.
// It reads the request, parses it, and sends the
appropriate response.
func handleConnection(conn net.Conn) {
    defer conn.Close()
    reader := bufio.NewReader(conn)
    request, err := reader.ReadString('\n')

    if err != nil {
        fmt.Println(err)
        return
    }
    fmt.Println("Received request: ", request)

    requestLine := strings.Fields(request)

    if len(requestLine) < 2 {
        fmt.Println("Invalid request")
        sendNotFoundResponse(conn)
        return
    }
    method := requestLine[0]
    url := requestLine[1]
```

```go
   if method != "GET" {
       fmt.Println("Unsupported method")
       sendNotFoundResponse(conn)
       return
   }

   // Send the appropriate response based on the
requested URL.
   if url == "/index.html" {
       sendHTMLResponse(conn, "index.html")
   } else if strings.HasSuffix(url, ".jpg") {
       sendFileResponse(conn, url[1:], "image/jpeg")
   } else if strings.HasSuffix(url, ".mp3") {
       sendFileResponse(conn, url[1:], "audio/mpeg")
   } else if strings.HasSuffix(url, ".ico") {
       sendFileResponse(conn, url[1:], "image/x-icon")
   } else {
       sendNotFoundResponse(conn)
   }
}

// sendHTMLResponse sends an HTML response to the client.
// It reads the specified file and sends it as the
response body.
func sendHTMLResponse(conn net.Conn, filename string) {
   content, err := os.ReadFile(filename)
   if err != nil {
       sendNotFoundResponse(conn)
       return
   }
   response := "HTTP/1.1 200 OK\r\n" +
        "Content-Type: text/html\r\n" +
        "Content-Length: " + strconv.Itoa(len(content)) +
"\r\n" +
        "Connection: close\r\n" +
        "\r\n" + string(content)
    conn.Write([]byte(response))
}

// sendFileResponse sends a file response to the client.
// It reads the specified file and sends it as the
response body.
func sendFileResponse(conn net.Conn, filename string,
contentType string) {
    content, err := os.ReadFile(filename)
    if err != nil {
        sendNotFoundResponse(conn)
        return
    }

    response := "HTTP/1.1 200 OK\r\n" +
        "Content-Type: " + contentType + "\r\n" +
        "Content-Length: " + strconv.Itoa(len(content)) +
"\r\n" +
        "Connection: close\r\n" +
        "\r\n"
    conn.Write([]byte(response))
    conn.Write(content)
}

// sendNotFoundResponse sends a 404 Not Found response to
the client.
func sendNotFoundResponse(conn net.Conn) {
    response := "HTTP/1.1 404 Not Found\r\n" +
        "Content-Type: text/plain\r\n" +
        "Content-Length: 13\r\n" +
        "Connection: close\r\n" +
        "\r\n" +
        "404 Not Found"
    conn.Write([]byte(response))
}
```

Brief Description

- Listening on Port 9999: The server listens for incoming TCP connections on port 9999.
- Accepting Connections: When a connection is accepted, it is handled in a new goroutine.
- Reading Requests: The server reads the incoming request from the connection.
- Parsing Requests: The request is parsed to extract the HTTP method and URL.
- Handling GET Requests: The server handles GET requests by checking the URL and responding with the appropriate file or a 404 error if the file is not found or the method is unsupported.
- Sending Responses: The server sends HTML, JPEG, MP3, or ICO files as responses based on the URL, or a 404 Not Found response if the file is not available or the request is invalid.

Code Flow

1. Start the Server
    - The main function initializes the server to listen on TCP port 9999.
    - net.Listen("tcp", ":9999") starts listening on port 9999.
    - A loop (for { ... }) continuously accepts new connections.
2. Accept Connections
    - ln.Accept() accepts an incoming connection.
    - A new goroutine (go handleConnection(conn)) is started to handle the connection, allowing the server to handle multiple connections concurrently.
3. Handle Connection
    - The handleConnection function processes the connection:

        - Reads the request using bufio.NewReader(conn) and reader.ReadString('\n').
        - Splits the request line into components using strings.Fields(request).
4. Parse and Validate Request
    - Checks if the request is valid by ensuring it has at least two parts.
    - Extracts the HTTP method and URL from the request line.
    - If the method is not "GET", sends a 404 Not Found response.
5. Respond to GET Requests
    - Checks the URL to determine the type of file requested:
        - If the URL is /index.html, calls sendHTMLResponse(conn, "index.html") to send the HTML file.
        - If the URL ends with .jpg, .mp3, or .ico, calls sendFileResponse(conn, url[1:], contentType) with the appropriate content type.
        - If the URL does not match any known file types, sends a 404 Not Found response.
6. Send HTML Response
    - The sendHTMLResponse function reads the specified HTML file and constructs an HTTP response.
    - Sends the response back to the client using conn.Write([]byte(response)).
7. Send File Response
    - The sendFileResponse function reads the specified file and constructs an HTTP response with the appropriate content type.
    - Sends the response and the file content back to the client using conn.Write([]byte(response)) and conn.Write(content).
8. Send 404 Not Found Response
    - The sendNotFoundResponse function constructs a 404 Not Found response and sends it to the client using conn.Write([]byte(response)).

Summary of Main Functions

- main: Sets up the server, listens on port 9999, and accepts connections.
- handleConnection: Reads and parses the request, and determines the appropriate response.
- sendHTMLResponse: Sends an HTML file as a response.
- sendFileResponse: Sends a file with a specified content type as a response.
- sendNotFoundResponse: Sends a 404 Not Found response for invalid requests or unsupported methods.