```c
/*
* Name: Rusho Binnabi
* Date: 04/29/2022
* Course Code and Title: ICSI 333 - System Fundamentals
* Semester: Spring 2022
* TA Name: Sourav Dutta
* Student ID: 001427521
*/

/* The work for this assignment was divided between me and my partner/teammate Rockwon Seo.
* I will be doing either the client side or server side of the program depending on which side my partner decides to work on.
* We collaborated on this numerous times over email.
* We both tried to create our own server and or client first then met and exchanged files via email and we helped each other improve it so to speak.
* The testing process was a similar process where we tried to test it on our own first and then met and exchanged files via email and we helped each other improve it in a sense.
*/

/*
* Unfornately, we tried the best we could but we unfortunately couldn't get the client working.
*/

#include <stdio.h> // includes the stdio library for use in this program.
#include <stdlib.h> // includes the stdlib library for use in this program.
#include <string.h> // includes the string library for use in this program.
#include <errno.h> // includes the errno library for use in this program.
#include <unistd.h> // includes the uninstd for use in this program.
#include <netdb.h> // includes the netdb for use in this program.
#include <sys/types.h> // includes the types library from sys for use in this program.
#include <sys/socket.h> // includes the socket library from sys in this program.
#include <netinet/in.h> // includes the in library from netinet in this program.
#include <arpa/inet.h> // includes the inet library from arpa in this program.

#define SIZE 1024 // defines a constant called SIZE with a value of 1024.
#define BACKLOG 10 // defines a constant called BACKLOG with a value of 10.

/* this program will act as the server side of the web server. */
/* the server will run in an infinite loop waiting for and process requests from the client. */

void serverReport(struct sockaddr_in *serverAddress); /* creates a void function called serverReport() with one argument which is a structure pointer sockaddr_in called serverAddress. */
                                                      /* this function creates and has the functionality for the server. */

void setHttpHeader(char header[]) { /* creates a void function called setHttpHeader() with one argument that is a char array called header. */
                                    /* this function sets the header for the http server. */

    FILE *htmlFileData = fopen("index.html", "r"); // creates a file pointer called htmlFileData which, by using the fopen() function, opens a html file called fileData in read mode.

    char lines[100]; // creates a char array called lines with a size of 100.
    char responseData[8000]; // creates a char array called responseData with a size of 8000.
    while (fgets(lines, 100, htmlFileData) != 0) { // the code in the while loop will run as long as, using the fgets() function and lines, a number of 100, and htmlFileData, is not equal to 0.
        strcat(responseData, lines); // using the strcat() function, it concatenates the string from lines into responseData.
    } // the end of the while loop.
    strcat(header, responseData); // using the strcat() function, it concatenates the string from responseData into header.
    fclose(htmlFileData); // using the fclose() function, it closes the file that's associated with htmlFileData.
} // the end of the setHttpHeader() function.

int main(void) { /* this int main() function with a void argument has the code needed for my program to run. */
    char header[8000] = "HTTP/1.0 200 OK\r\n\n"; // creates a char array called header with a size of 8000 and is initialized with a string message of "HTTP/1.0 200 OK\r\n\n".
    int serverSocket = socket(AF_INET, SOCK_STREAM, 0); // creates an integer variable called serverSocket which, using the socket() function, will have the values of the arguments AF_INET, SOCK_S

    struct sockaddr_in serverAddress; // creates a structure sockaddr_in called serverAddress.

    serverAddress.sin_family = AF_INET; // using sin_family, it assigns the value of AF_INET to serverAddress.
    serverAddress.sin_port = htons(8000); // using sin_port, it assigns the value of 8000 using the htons() function which is the port number.
    serverAddress.sin_addr.s_addr = htonl(INADDR_LOOPBACK); // using sin_addr.s_addr, it assigns the value of the argument INADOR_LOOPBACK to serverAddress form the htonl() function.

    bind(serverSocket, (struct sockaddr *) &serverAddress, sizeof(serverAddress)); // using the bind() function, it binds serverSocket, the address of serverAddress using the size of serverAddress.

    int listenforRequest = listen(serverSocket, BACKLOG); // creates an integer variable called listenForRequest which will have the values of the arguments serverSocket and BACKLOG from the listen() fu

    if (listenforRequest < 0) { // the code inside the if statement will run if the value inside listenForRequest is less than 0.
        printf("Server error. The server could not listen for and process requests."); fflush(stdout); // tells the user that there was an error with the server and could not listen for and process requests. Then it f
        return 1; // returns a 1 which means something went wrong.
    } // the end of the if statement.

    serverReport(&serverAddress); // calls the severReport() function using the address of serverAddress.

    setHttpHeader(header); // calls the setHttpHeader() function using header.

    int client; // creates an integer variable called client.

    while (1) { // creates an infinite loop using the value of 1 that processes and waits for requests from a client.
        client = accept(serverSocket, NULL, NULL); // client will have the value of the arguments serverSocket, NULL, and NULL from the accept() function.
        send(client, header, sizeof(header), 0); // using the send() function, it sends the data from client, header, by using the size of header, and a value of 0.
        close(client); // using the close() function, it closes client.
    } // the end of the infinite loop.
} // the end of the int main() function.

void serverReport(struct sockaddr_in *serverAddress) { /* creates a void function called serverReport() with one argument which is a structure pointer sockaddr_in called serverAddress. */
                                                       /* this function creates and has the functionality for the server. */
    char bufferForHost[INET6_ADDRSTRLEN]; // creates a char array called bufferForHost which will have the size of INET6_ADDRSTRLEN.
    char serviceBuffer[NI_MAXSERV]; // creates a char array called serviceBuffer which will have the size of NI_MAXSERV.
    socklen_t addr_len = sizeof(*serverAddress); // creates a socklen_t called addr_len which will have the size of serverAddress as a pointer.

    int error = getnameinfo((struct sockaddr *) serverAddress, addr_len, bufferForHost, sizeof(bufferForHost), serviceBuffer, sizeof(serviceBuffer), NI_NUMERICHOST); // creates an integer variable calle
                                                                                                                                                                       //  structure pointer called serverAddress, addr_len, bufferForHost, the size of bufferForHost,
                                                                                                                                                                       // serviceBuffer, the size of serviceBuffer, and NI_NUMERICHOST from the getnameinfo() function.

    if (error != 0) { // if the value of error is not equal to 0, then the code inside the if statement will be run.
        printf("The server is not working.\n"); fflush(stdout); // tells the user that the server is not working. Then it flushes the buffer by using the fflush() function and stdout as it's argument.
    } // the end of the if statement.

    printf("\nThe server is listening on http://%s:%s\n", bufferForHost, serviceBuffer); // tells the user that the server is listening on the specified string arguments of bufferForHost and serviceBuffer.

} // the end of the serverReport() function.
```