```
// Group G
// Cody Ray
// cody.a.ray@okstate.edu
// 10/10/2022
// This file inlcudes the server and client code as well as code for a simple
main menu to handle user input

//SERVER----------------------------------------------------------------------
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <stdbool.h>

int main(){
    //port used for connection between client and server
    #define port 4567
    //structure that allows the storage of different values for the server
address
    struct sockaddr_in serverAddress;
    //buffer that stores input from client to server and messages from server to
client
    char buffer[512];
    //initializes the server socket and sets it up to be an ipv4 TCP socket with
no protocols
    int serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    //checks to make sure the socket was created successfully
    if(serverSocket < 0){
        printf("Error Creating Server Socket\n");
        return 0;
    }
    printf("Server Socket Created Successfully\n");

    //sets the server to an ipv4 socket
    serverAddress.sin_family = AF_INET;
    //ensures the server port value is stored correctly by using htons
    //which takes 16-bit host byte numbers and returns the 16-bit numbers in
network byte order
    serverAddress.sin_port = htons(port);
    //sets the server address to 127.0.0.1 which is the ip for local host
    serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1");

    //tries to bind the server to the socket using the server address
    int serverBind = bind(serverSocket, (struct sockaddr*) &serverAddress,
sizeof(serverAddress));
    //checks to make sure the server was able to bind to the socket successfully
    if(serverBind < 0){
        printf("Error Binding Server To Socket\n");
        return 0;
    }
    printf("Server Bound To Port %d\n", port);

    //Starts listening for connections on the socket we created for the server
    listen(serverSocket, 3);
    printf("Waiting For Connections...\n");

    //new values to store information for each of the child servers that are
going to handle the
    //interactions with each new client
    int newSocket;
    struct sockaddr_in newServerAddress;
    socklen_t newServerAddressSize;
    pid_t childProcessID;

    //infinite loop to handle the rest of the server client connections and
communication
    while(1){
        //creates a new server socket for future clients to connect to and sends
an error message if something is wrong
        newSocket = accept(serverSocket, (struct sockaddr*)&newServerAddress,
&newServerAddressSize);
        if(newSocket < 0){
            printf("Error Creating Child Socket\n");
            return 0;
        }
        printf("Connection made on the IP %s and port %d\n",
inet_ntoa(newServerAddress.sin_addr), ntohs(newServerAddress.sin_port));

        //creates each of the child processes
        if((childProcessID = fork()) == 0){
            //closes the main port that clients use to connect in the child
processes
            close(serverSocket);
            //variables to make the main menu work and store user input
            char filename[36];
            char column[36];
            char uniqueValue[128];
            bool mainMenu = true;
            int currentRecievedMsg = 0;

            //while loop that handles the main menu
            while(mainMenu){
                //loop which handles telling the client what to input
                bzero(buffer,sizeof(buffer));
                if(currentRecievedMsg == 1){
                    send(newSocket, "Choose an option 1.bookInfo.txt or 
2.amazonBestsellers.txt\n", strlen("Choose an option 1.bookInfo.txt or 
2.amazonBestsellers.txt\n"),0);
                    bzero(buffer,sizeof(buffer));
                }else if(currentRecievedMsg == 2){
                    send(newSocket, "Choose an option 1.Book category or 2.Star 
rating or 3.Stock\n", strlen("Choose an option 1.Book category or 2.Star rating 
or 3.Stock\n"),0);
                    bzero(buffer,sizeof(buffer));
                }else if(currentRecievedMsg == 3){
                    send(newSocket, "Choose an option 1.User rating or 2.Year or 
3.Genre\n", strlen("Choose an option 1.User rating or 2.Year or 3.Genre\n"),0);
                    bzero(buffer,sizeof(buffer));
                }

                //receives messages from the client
                recv(newSocket, buffer, 512, 0);
                //checks to make sure the client has not exited and if it has it 
disconnects the child process and closes the socket
                if(strcmp(buffer, "exit") == 0){
                    close(newSocket);
                    printf("Disconnect on the IP %s and port %d\n", 
inet_ntoa(newServerAddress.sin_addr), ntohs(newServerAddress.sin_port));
                    return 0;
                }else if(strlen(buffer) > 0){
                    //displays what the client has sent to the server
                    printf("Client sent: %s\n", buffer);
                }

                //the first loop after random client input
                if(currentRecievedMsg == 0){
                    currentRecievedMsg = 1;
                    printf("currentRecievedMSG = 1\n");
                //loop that handles choosing which files to get options from
                }else if(currentRecievedMsg == 1){
                    //sets the filename value to bookinfo.txt if the user input
was 1
                    if(strcmp(buffer, "1") == 0){
                        strcpy(filename,"bookInfo.txt");
                        currentRecievedMsg = 2;
                        printf("Input was 1 and filename set to bookInfo.txt new
currentRecievedMSG = 2\n");
                    //sets the filename to amazonbestsellers.txt if the user
input was 2
                    }else if(strcmp(buffer, "2") == 0){
                        strcpy(filename,"amazonBestsellers.txt");
                        currentRecievedMsg = 3;
                        printf("Input was 2 and filename set to
amazonBestsellers.txt new currentRecievedMSG = 3\n");
                    }
                //loop that handles the user options for bookinfo.txt
                }else if(currentRecievedMsg == 2){
                    //sets the column value to book category if the user input
was 1
                    if(strcmp(buffer, "1") == 0){
                        strcpy(column,"Book category\n");
                        printf("Input was 1 and column set to book category new
currentRecievedMSG = 4\n");
                        printf("Main menu completed\n");
                        mainMenu = false;
                    //sets the column value to star rating if the user input was
2
                    }else if(strcmp(buffer, "2") == 0){
                        strcpy(column,"Star rating\n");
                        printf("Input was 2 and column set to star rating new
currentRecievedMSG = 4\n");
                        printf("Main menu completed\n");
                        mainMenu = false;
                    //sets the column value to stock if the user input was 3
                    }else if(strcmp(buffer, "3") == 0){
                        strcpy(column,"Stock\n");
                        printf("Input was 3 and column set to stock new
currentRecievedMSG = 4\n");
                        printf("Main menu completed\n");
                        mainMenu = false;
                    }
                //loop that handles the user options for amazonbestsellers.txt
                }else if(currentRecievedMsg == 3){
                    //sets the column value to user rating if the user input was
1
                    if(strcmp(buffer, "1") == 0){
                        strcpy(column,"User rating\n");
                        printf("Input was 1 and column set to user rating new
currentRecievedMSG = 4\n");
                        printf("Main menu completed\n");
                        mainMenu = false;
                    //sets the column value to year if the user input was 2
                    }else if(strcmp(buffer, "2") == 0){
                        strcpy(column,"Year\n");
                        printf("Input was 2 and column set to year new
currentRecievedMSG = 4\n");
                        printf("Main menu completed\n");
                        mainMenu = false;
                    //sets the column value to genre if the user input was 3
                    }else if(strcmp(buffer, "3") == 0){
                        strcpy(column,"Genre\n");
                        printf("Input was 3 and column set to genre new
currentRecievedMSG = 4\n");
                        printf("Main menu completed\n");
                        mainMenu = false;
                    }
                }
            }

            //prints the users input and sends the client confirmation that the
main menu is completed
            printf("%s\n", filename);
            printf("%s\n", column);
            bzero(buffer,sizeof(buffer));
            send(newSocket, "Main Menu Input Completed\n", strlen("Main Menu
Input Completed\n"), 0);
            bzero(buffer,sizeof(buffer));

            while(1){
                //receives messages from the client
                recv(newSocket, buffer, 512, 0);
                //checks to make sure the client has not exited and if it has it
disconnects the child process and closes the socket
                if(strcmp(buffer, "exit") == 0){
                    close(newSocket);
                    printf("Disconnect on the IP %s and port %d\n",
inet_ntoa(newServerAddress.sin_addr), ntohs(newServerAddress.sin_port));
                    return 0;
```

```c
                }else if(strlen(buffer) > 0){
                    //displays what the client has sent to the server
                    printf("Client sent: %s\n", buffer);
                    //sends the message back to the client
                    send(newSocket, buffer, strlen(buffer), 0);
                    //resets the buffer to all null values
                    bzero(buffer,sizeof(buffer));
                }
            }
        }
    }
    return 0;
}
//END OF SERVER----------------------------------------------------

//CLIENT-----------------------------------------------------------
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(){
    //port used for connection between client and server
    #define port 4567
    //structure that allows the storage of different values for the server
address
    struct sockaddr_in serverAddress;
    //buffer that stores input from client to server and messages from server to
client
    char buffer[512];
    //initializes the client socket and sets it up to be an ipv4 TCP socket with
no protocols
    int clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    //checks to make sure the socket was created successfully
    if(clientSocket < 0){
        printf("Error Creating Client Socket\n");
        return 0;
    }
    printf("Client Socket Created Successfully\n");

    //sets the server to an ipv4 socket
    serverAddress.sin_family = AF_INET;
    //ensures the server port value is stored correctly by using htons
    //which takes 16-bit host byte numbers and returns the 16-bit numbers in
network byte order
    serverAddress.sin_port = htons(port);
    //sets the server address to 127.0.0.1 which is the ip for local host
    serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1");

    //tries to connect to the server using the clients socket and the server
address
    int serverConnect = connect(clientSocket, (struct sockaddr*) &serverAddress,
sizeof(serverAddress));
    //checks to make sure the client was able to connect to the server
successfully
    if(serverConnect < 0){
        printf("Unable To Connect To Server\n");
        return 0;
    }
    printf("Successfully Connected To Server\n");

    //resets the buffer to all null values
    bzero(buffer,sizeof(buffer));

    //infinite loop which handles the rest of the client server connections
    while(1){
        //scans user input and sends it to the server
        printf("Client: ");
        scanf("%s", &buffer[0]);
        send(clientSocket, buffer, strlen(buffer), 0);

        //checks if there are errors in input sent by server and displays input
if there are no issues
        if(recv(clientSocket, buffer, 512, 0) < 0){
            printf("Error In Message Sent to Server");
        }else{
            printf("%s\n", buffer);
        }

        //ends the connection with the server if exit is inputted
        if(strcmp(buffer, "exit") == 0){
            return 0;
        }
    }
    return 0;
}
//END OF CLIENT-----------------------------------------------------------
```