```
/*

Group: B

Name: Brennan Schlittler

Email: brennan.schlittler@okstate.edu

Date: 10/10/22

Description:

Starts a server and handles cient connections and communication


compile: gcc -Wall server.c process.c -lrt -o server

execute: ./server

Tested on csx2

*/


#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#include <errno.h>



#include "process.h"


int main()

{

   /Initialize values
```

```
int port = 5150;

int serverSock, clientSock;

struct sockaddr_in serverAddress, clientAddress;

socklen_t addrSize;

char buffer[1024];

pid_t childPid;


//create socket, error if it fails

serverSock = socket(AF_INET, SOCK_STREAM, 0);

if(serverSock< 0)

{

   perror("[-] Error creating socket");

   exit(1);

}

printf("[+] TCP socket created successfully \n");


//Create buffer and update address

memset(buffer, '\0', sizeof(buffer));

memset(&serverAddress, '\0', sizeof(serverAddress));

serverAddress.sin_family = AF_INET;

serverAddress.sin_port = htons(port);

serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1");


//Bind scoket, error if it fails

int bindingSuccess = bind(serverSock, (struct sockaddr*)&serverAddress, sizeof(serverAddress));

if(bindingSuccess < 0)

{

   perror("[-] Error in binding");

   exit(1);
```

```
}



//Wait for client to connect
listen(serverSock, 3);
printf("Waiting for connection \n");
int numClients = 0;

while (1) {

  //Accept client connection
  clientSock = accept(serverSock, (struct sockaddr*)&clientAddress,&addrSize);
  if (clientSock < 0) {
    exit(1);
  }

  if ((childPid = fork()) == 0) {
    //Go into child process
    close(serverSock);

    //previously use #define MAX 200, in the #include section of the file
    FILE *filePointer ;
    char data[100];
    char options[100][100];
    filePointer = fopen("options.txt", "r") ; //Opening the Options.txt file

    if(filePointer == NULL)
    {
        printf("File cant be opened");
```

```
            exit(0);
        }
        else
        {
          int i=0;
          while(fgets(data,100,filePointer)!=NULL) //reading the file line - by - line
          {
                strcpy(options[i],data); //storing it in the options array. and the first element in the array
contains the options that needs to be sent to the client
                i++;
          }
        }

        printf("Sending options to the client \n");
        send(clientSock, options[0], sizeof(options[0]), 0);  //sending the options to the client
        printf("Waiting for the Client feedback \n");

        // read the message from client and copy it in buffer
        bzero(buffer, sizeof(buffer));
        recv(clientSock, buffer, 1024, 0);
        printf("Response: %s \n\n", buffer);

        //Create Pipes
        int fin[2];
        int fout[2];
        if(pipe(fin) < 0 || pipe(fout) < 0){
          perror("pipe");
          exit(1);
        }
```

```
// print buffer which contains the client contents
int option = atoi(buffer);
if(option == 1){

    char column_names[50] = "Book category,Star rating,Stock";

    write(clientSock,column_names, sizeof(column_names));
    printf("Waiting for column options\n");

    bzero(buffer, sizeof(buffer));
    recv(clientSock, buffer, 1024, 0);
    printf("%s\n",buffer);

    option = atoi(buffer);
    readFile("bookInfo.txt", 6);
    if(option == 1)
        processSetup(703, 6, 1, 43, fin, fout);
    else if(option == 2)
        processSetup(703, 6, 2, 5, fin, fout);
    else if(option == 3)
        processSetup(703, 6, 4, 2, fin, fout);
    else
        printf("Incorrect category.\n");

}
else if(option == 2){
    char column_names[50] = "User rating, Year, Genre";
    write(clientSock, column_names, sizeof(column_names));
    printf("Waiting for column options\n");

    bzero(buffer, sizeof(buffer));
    recv(clientSock, buffer, 1024, 0);
    printf("Response: %s\n",buffer);
    option = atoi(buffer);

    readFile("amazonBestsellers.txt", 7);
    if(option == 1)
        processSetup(550, 7, 2, 10, fin, fout);
    else if(option == 2)
        processSetup(550, 7, 5, 11, fin, fout);
    else if(option == 3)
        processSetup(550, 7, 6, 2, fin, fout);
    else
        printf("Incorrect category.\n");

}

//Close unneccesary pipe ends
close(fin[1]);
close(fout[0]);

//wait for processes to finish
char wait_buff[10];
read(fin[0], wait_buff, 10);
send(clientSock, "Ready", 10, 0);
```

```
char process_buff[4000];
//Menu Loop
while(1){
    //recieve choice from user
    bzero(buffer, sizeof(buffer));
    memset(buffer, 0 , sizeof(buffer));
    recv(clientSock, buffer, 1024, 0);
    int option = atoi(buffer);

    //write choice to process
    size_t length = strlen( buffer );
    write( fout[1], buffer, length );

    //read response from process
    bzero(process_buff, sizeof(process_buff));
    read(fin[0], process_buff, 4000);

    //send result from process to client
    send(clientSock, process_buff, sizeof(process_buff), 0);

    if(option == 1){
        //wait for process list
        bzero(buffer, sizeof(buffer));
        recv(clientSock, buffer, 1024, 0);

        //write client choice to process
        length = strlen(buffer);
        write(fout[1], buffer, length);
```

```
                //read string array from process
                char arr[200][200];
                if(read(fin[0], arr, sizeof(sizeof(char) * 200) * 200) < 0){
                    return 1;
                }

                //write string array to client
                if(write(clientSock, arr, sizeof(sizeof(char) * 200) * 200) < 0){
                    return 3;
                }

            }
            else if(option > 3 && option < 1){
                break;
            }

        }



    }
    close(clientSock);
    numClients--;
  }
  // Close the client socket id
  close(clientSock);
  return 0;
}
```