```
/
*
    * Created by: Maxwell Meckling
    * Train Seating is designed to help manage everything to do with seating on the
    train.
    * This includes the creation of new seats each day through a struct.
    */



    #include "max_trainSeating.h" //Include our own header file


    //calculates and then returns the number of available seats for a given day
    int countNumberOfAvailableSeats(availableSeats *dayToCount) {
       int numberOfAvailableSeats = 0;
       for(int i = 0; i < sizeof(dayToCount->seats) / sizeof(int); i++) {
           if(dayToCount->seats[i] == 0) {
               numberOfAvailableSeats++;
           }
       }
       return numberOfAvailableSeats;
    }


    //Returns 0 or 1 based on the passed dayOfTravel variable to match which day we
    should be on.
    //Looks at the int, not the string
    int matchDayOfTravel(availableSeats *ptr, int dayOfTravel) {
       //if statement to compare which dayOfTravel we are on
       if((ptr)->dateInt == dayOfTravel) {
           //day 1 match
```

```
        return 0;
    } else {
        //day 2 match
        return 1;
    }
}


//sends a message to the client through tcp
void seatingSendMessageToClient(char *message, int socket){
        char stringBuffer[STRING_BUFFER_MAX];
        strcpy(stringBuffer,message);
        send(socket,stringBuffer,sizeof(stringBuffer),0);
}




//accesses shared memory struct member .nextTicketNumber to assign next
available ticket number to customer
//then increments ticket number for next customer
int assignTicketNumber(customerInfo nextCustomer, int socket, availableSeats
*ptr){
    printf("\nassignTicketNumber() called\n"); //for debugging


    //Variable to help match the day we are on to the proper struct in the
shared memory pointer object: (ptr+currentDayModifier)
    int currentDayModifier = matchDayOfTravel(ptr, nextCustomer.dayOfTravel);


    //Get the nextTicketNumber from shared memory
    int nextTicketNumber = (ptr+currentDayModifier)->ticketNumber;


    //Update the ticketNumber in shared memory (just increment by 1)
    (ptr+currentDayModifier)->ticketNumber =
(ptr+currentDayModifier)->ticketNumber + 1;
```

```
       //Send a message to the client to let them know what their ticket number
is.
   char messageToPassToClient[100] = "";
   sprintf(messageToPassToClient, "\nYour ticket number is: %d",
nextTicketNumber);
   seatingSendMessageToClient(messageToPassToClient, socket);


   return nextTicketNumber;
}



//Checks to make sure there is a seat available for the client based on the
numberOfTravelers they requested.
//Returns true if there are seats available and false if there aren't seats
available.
//Also displays a message to the client if there aren't enough seats available.
bool checkIfAvailableSeats(int dayOfTravel, int numberOfTravelers, int socket,
availableSeats *ptr){
   printf("\ncheckIfavailableSeats() called\n"); //for debugging


   //Variable to help match the day we are on to the proper struct in the
shared memory pointer object: (ptr+currentDayModifier)
   int currentDayModifier = matchDayOfTravel(ptr, dayOfTravel);


   int numberOfAvailableSeats =
countNumberOfAvailableSeats((ptr+currentDayModifier));


   if(numberOfTravelers > numberOfAvailableSeats) {
       //more travelers than available seats
       char messageToPassToClient[100] = "";
       sprintf(messageToPassToClient, "\nSorry, but there aren't %d seats
available right now. We only have %d seats open.", numberOfTravelers,
numberOfAvailableSeats);
       seatingSendMessageToClient(messageToPassToClient, socket);
       return false;
   } else {
       //equal to or more than enough seats for the number of travelers
       return true;
   }
}



//shows seats customer selects starting index (seat) and #of travelers fills in
seats
//accesses shared memory to read seats available and copies to string buffer
and then sends to client via tcp
void displayAvailableSeats(int dayOfTravel, int numberOfTravelers, int socket,
availableSeats *ptr){
       printf("\ndiplayAvailalbeSeats() called\n"); //for debugging


   //Variable to help match the day we are on to the proper struct in the
shared memory pointer object: (ptr+currentDayModifier)
   int currentDayModifier = matchDayOfTravel(ptr, dayOfTravel);


   char messageOnEachRow[100] = "";
   char messageToPassToClient[300] = "";


   //Backend looking display
   for(int i = 0; i < 3; i++) { //i < sizeof(currentDay.seats) / sizeof(int) /
9;
       sprintf(messageOnEachRow, "%d:%d, %d:%d, %d:%d, %d:%d, %d:%d, %d:%d,
%d:%d, %d:%d, %d:%d \n",
               i, (ptr+currentDayModifier)->seats[i], i+3,
(ptr+currentDayModifier)->seats[i+3], i+6,
(ptr+currentDayModifier)->seats[i+6],
               i+9, (ptr+currentDayModifier)->seats[i+9], i+12,
(ptr+currentDayModifier)->seats[i+12], i+15,
(ptr+currentDayModifier)->seats[i+15],
               i+18, (ptr+currentDayModifier)->seats[i+18], i+21,
(ptr+currentDayModifier)->seats[i+21], i+24,
(ptr+currentDayModifier)->seats[i+24]);
       strcat(messageToPassToClient, messageOnEachRow); //save each row onto
the main message
   }
   seatingSendMessageToClient(messageToPassToClient, socket);
}
```

```
//Allows the client to select their seats on the train.

//This is done by asking the to select each seat individually up to the amount
of seats they specified on their ticket in the addedSeatsIfModified variable.
//Returns the customerInfo struct with the client's selected seats and updates
those seats in shared memory too.
customerInfo selectAvailableSeats(customerInfo nextCustomer,int socket,int
addedSeatsIfModified, availableSeats *ptr) {
   printf("\nselectAvailalbeSeats() called\n"); //for debugging

   char stringBuffer[STRING_BUFFER_MAX];


   //When there are actual seats to change

   seatingSendMessageToClient("\nWelcome to seat selection.", socket);

   while(addedSeatsIfModified == 0) {

       seatingSendMessageToClient("\nHow many seats would you like to select?:
", socket);


       //receive response via tcp

           strcpy(stringBuffer,"needint"); //code that customer will read and no
to scanf for int
           send(socket,stringBuffer,sizeof(stringBuffer),0);

           recv(socket,&addedSeatsIfModified,sizeof(int),0); //change the
addedSeatsIfModified variable to match how many seats the user wants


       //check to make sure we have enough seats available based on what the
user just entered.
       if(!checkIfAvailableSeats(nextCustomer.dayOfTravel,
addedSeatsIfModified, socket, ptr)) {
             addedSeatsIfModified = 0;

       }

   }


   //Declare variable to use

   int currentSelectedSeatNumber;
```

```
    //Variable to help match the day we are on to the proper struct in the
shared memory pointer object: (ptr+currentDayModifier)
   int currentDayModifier = matchDayOfTravel(ptr, nextCustomer.dayOfTravel);



   //Begin gathering user input

   seatingSendMessageToClient("\nFor the following prompt(s), please enter an
integer (value 0 to 26) matching an available seat from above.", socket);
   seatingSendMessageToClient("\nA seat is available if it has a 0 next to it.
So for example, 6:0 would be open while 6:1 would be taken.\n", socket);
   for(int i = 0; i < addedSeatsIfModified; i++) {

       char messageToPassToClient[100] = "";

       sprintf(messageToPassToClient, "\nPlease select seat %d out of %d: ",
i+1, addedSeatsIfModified);
       seatingSendMessageToClient(messageToPassToClient, socket);


       //receive response via tcp

           strcpy(stringBuffer,"needint"); //code that customer will read and no
to scanf for int
           send(socket,stringBuffer,sizeof(stringBuffer),0);

           recv(socket,&currentSelectedSeatNumber,sizeof(int),0); //If the user
doesn't enter an integer, an error will be thrown here.


       if(currentSelectedSeatNumber < 0 || currentSelectedSeatNumber > 26) {

           //catch possible problem of user entering an int outside our scope
of seat numbers
           seatingSendMessageToClient("\nError: Please try again and enter a
seat number between 0 and 26.", socket);
           //lower i by 1 so that the user has a chance to try again

           i--;

       } else if((ptr+currentDayModifier)->seats[currentSelectedSeatNumber] ==
1) {
           //if the seat is already selected then the user will have to select
another one
           seatingSendMessageToClient("\nError: Seat already taken. Please
select an open seat.", socket);
           //lower i by 1 so that the user has a chance to try again

           i--;
       } else {

           //otherwise update the customer's booked seats array with their
selected seat
           nextCustomer.bookedSeats[currentSelectedSeatNumber] = 1;

           //Modify the seats in to be later put into shared memory too!!!

           (ptr+currentDayModifier)->seats[currentSelectedSeatNumber] = 1;

       }

   }

   return nextCustomer;

}



//Frees either all of the passed customerInfo struct's seats from the struct
and from shared memory or frees a client specified amount.
//If the user wants to reduce their seats by a specific amount, then they will
be asked to individually select which seats they no longer want up to that
amount.
customerInfo freeCustomersSeatsInSharedMem(customerInfo customerMods, int
socket, int customersRequestedSeatReduction, availableSeats *ptr) {
   printf("\nfreeCustomersSeatsInSharedMem() called\n"); //for debugging

   char stringBuffer[STRING_BUFFER_MAX];


   //Variable to help match the day we are on to the proper struct in the
shared memory pointer object: (ptr+currentDayModifier)
   int currentDayModifier = matchDayOfTravel(ptr, customerMods.dayOfTravel);


   if(customersRequestedSeatReduction != 0) {


       //Declare variables to use

       int currentSelectedSeatNumber;


       //Ask customer which seats they would like to free specifically

       seatingSendMessageToClient("\nFor the following prompt(s), please enter
an integer (value 0 to 26) of a seat you have already booked from above.\n",
socket);
       for(int i = 0; i < customersRequestedSeatReduction; i++) {
              char messageToPassToClient[100] = "";
           sprintf(messageToPassToClient, "\nWhich seat would you like to free
next? (freeing seat %d of %d): ", i+1, customersRequestedSeatReduction);
           seatingSendMessageToClient(messageToPassToClient,socket);


           //receive response via tcp
                 strcpy(stringBuffer,"needint"); //code that customer will read
and no to scanf for int
                 send(socket,stringBuffer,sizeof(stringBuffer),0);
                 recv(socket,&currentSelectedSeatNumber,sizeof(int),0);


           if(currentSelectedSeatNumber < 0 || currentSelectedSeatNumber > 26)
{
               //catch possible problem of user entering an int outside our
scope of seat numbers
               seatingSendMessageToClient("\nError: Please try again and enter
a seat number between 0 and 26.", socket);
               //lower i by 1 so that the user has a chance to try again
               i--;
           } else if(customerMods.bookedSeats[currentSelectedSeatNumber] == 0)
{
               //if the user selects a seat they don't own, then they will have
to try again and select one they do own
               char messageToPassToClient[100] = "";
              sprintf(messageToPassToClient, "\nError: You don't own seat %d.
Please select a seat you already have selected to remove.",
currentSelectedSeatNumber);
               seatingSendMessageToClient(messageToPassToClient, socket);
               //lower i by 1 so that the user has a chance to try again
               i--;
           } else {
               //otherwise reset the user selected seat
               customerMods.bookedSeats[currentSelectedSeatNumber] = 0;
               //Reset for shared memory too!!!
               (ptr+currentDayModifier)->seats[currentSelectedSeatNumber] = 0;
           }
```

```
        }

    } else {

        //reset bookedSeats completely for the customer and for shared memory
object
        for(int i = 0; i < 27; i++) {

            customerMods.bookedSeats[i] = 0;

            //Reset for shared memory too!!!

            (ptr+currentDayModifier)->seats[i] = 0;

        }

    }


    return customerMods;

}
```