C and UNIX Programming CS23710

Monitoring Runners and Riders

|  |  |
| --- | --- |
| Author: | James Upshall (jau1) |
|  |  |
| Date: | 14/12/2012 |
|  |  |
|  |  |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2012

CONTENTS

CONTENTS 2

1. Introductory Discussion 3

2. Source Code 4

2.1 Main.c 4

2.2 Structures.h 9

2.3 RaceDataHandler.c 11

2.4 NowThisIsPodRacing 14

3. Screenshot Tests 15

# Introductory Discussion

The initial problem of this assignment to me was that there were many data files that required storage and ordering so that you could process them, so I began my design with structures. I essentially decided upon one for each of the initial data files this created a structure in which all the data could be stored and accessed. For the event file I stored everything simple as arrays of characters as this data is not used for any processing and is useful to be used at the end to produce the results. Then I created the node structure this contained the identifying id number, as an integer, the type which I stored as an array of char, and an array of integers to model the nodes that you could travel to from any particular integer. The track similarly had and id and integers to represent the starting node and the time it would take to complete. For the courser I decided to store the actual course itself as an array of integers as this models the series nodes, I also created an array of tracks but I ended up not using this in my program. The entrant structure is a little more complex I split some of the data up to make it simpler to manage, independently storing hours and minutes as integers as no race would go past midnight and also the start times and a running total of the time taken by the rider.

I moved the majority of the ‘parsing’ code that interpreted the files into a separate file called RaceDataHandler.c I felt this kept a lot of unnecessary clutter form main.c. These functions all follow similar patterns in opening the data file that is given to them by a user and trying to order the data into the structures. I think this is one of the key features of C as a language it contains a lot of the lower level character manipulation that is lost in very high level languages. I think I was rather blasé about the lengths of the char arrays that I used in this section and in defining the structures whilst some were defined in the specification for the assignment, others felt quite arbitrary. The input of the data also lacks any validation or any kind of checking; whilst this isn’t necessarily a key requirement of the program assuming it is only used by race officials on a regular computer. In retrospect some of the iteration looks quite clumsy when moving through the elements of the structures.

After the data files are entered and stored the focus of the program shifts back to main.c in which I used a switch statement to create a little menu system for the user to select what they want to do. The first option is to find a competitor this takes in input from the keyboard before it is passed on to a function, which I stored in NowThisIsPodRacing.c, a file I again used to modularise some of the searching and functional code. To find a competitor I used a for loop to compare each of the names with the entrant structure files to that given by the user, if it is found some data is printed about that racer, if not then the program returns to the menu. The second options of finding those whom have not started is also stored in the NowThisIsPodRacing.c file this cycles through the entrants and increases and counter for each that haven’t finished, the option of finding those whom are on the course and those whom have finished is achieved in a similar way. By using the integer of stat in the entrant to represent states, 1 being have yet to start, 2 being currently racing and 3 being for those whom have finished.

The next possible instruction is to supply times manually, here I have switched back so sections of code in main.c, I am not entirely sure why. This choice works through a series of questions in which the data about an entrant is received from a user, that entrant is then updated with their new location, time and, status. After this is a similar choice to add times by a data file, this works similarly to that of the original data file inputs and is again the changing of the structure values based on the interpreted data. This section does include perhaps the most terrible piece of code that I have written though:

**if** **(** this\_course**[**this\_entrant**[**guy**].**course**].**nodes**[**this\_course**[**this\_entrant**[**guy**].**course**].**number\_of\_nodes**]**

**==** place **)** this\_entrant**[**guy**].**stat **=** 3**;**

This compares the final node value of the course to the current node value of the course but it is all relative to the entrant (guy) so that if the entrant is indeed in the last node of the course then they have finished and so changed their status as such.

The Final option before quit which just exits the program is that to print out the results. For this I have a simple heading and then the general information about the race day, after this a statement is printed about each entrant, I used a switch statement to so that there is different messages depending on whether the entrant had finished the race or not, I originally planned to use an else if statement but I could not find a similar syntax in C, and after this the switch statement seemed the simplest way to separate the types of results. Then a statement is printed out including the data on the entrants form the entrant structure.

# Source Code

## Main.c

/\*

\* File: main.c

\* Author: James Upshall (jau1)

\*

\* This is the main file for the Country Race project

\*

\*/

#include <stdio.h>

#include <stdlib.h>

#include "structures.h"

/\*

\*

\*/

int main**(**int argc**,** char**\*\*** argv**)** **{**

int command**;**

char name\_filename**[**31**];**

int Num\_Entrants**;**

int which**;**

struct event this\_event**;**

struct node this\_node**[**50**];**

struct track this\_track**[**50**];**

struct course this\_course**[**10**];**

struct entrant this\_entrant**[**31**];**

/\*this is the setup of all the files could do with a little validation\*/

printf**(**"where is the event file for this race held?\n"**);**

scanf**(** " %s"**,** name\_filename**);**

read\_in\_event**(**name\_filename**,** this\_event**);**

printf**(**"where is the node file for this race held?\n"**);**

scanf**(** " %s"**,** name\_filename**);**

read\_in\_nodes**(**name\_filename**,** this\_node**);**

printf**(**"where is the track file for this race held?\n"**);**

scanf**(** " %s"**,** name\_filename**);**

read\_in\_tracks**(**name\_filename**,** this\_track**,** this\_node**);**

printf**(**"where is the courses file for this race held?\n"**);**

scanf**(** " %s"**,** name\_filename**);**

read\_in\_courses**(**name\_filename**,** this\_course**);**

printf**(**"where is the entrants file for this race held?\n"**);**

scanf**(** " %s"**,** name\_filename**);**

read\_in\_entrants**(**name\_filename**,** this\_entrant**,** Num\_Entrants**);**

**do** **{**

printf**(**"What do you want to do?\n"**);**

printf**(**"%d=find a competitor, %d=find those competitors whom have yet to start,"

"%d=find out how many are on the course, %d=find out how many have finished"

"%d=add some new times manually %d=add some new times with a file "

"%d= print out the results table %d=Quit?,"**,**

FIND\_COMPETITOR**,** FIND\_COMPETITORS\_YET\_TO\_START**,**

FIND\_NUMBER\_ON\_COURSE**,** FIND\_HOW\_MANY\_FINISHED**,** SUPPLY\_TIMES**,**

READ\_IN\_FILE**,** PRODUCE\_RESULTS**,** QUIT**);**

scanf**(**" %d"**,** **&**command**);**

**switch** **(**command**)** **{**

**case** FIND\_COMPETITOR**:**

printf**(**"so who in particular were you looking for?\n"**);**

char competitor**[**51**];**

scanf**(**" %s"**,** competitor**);**

Find\_Him**(**competitor**,** Num\_Entrants**,** this\_entrant**);**

**break;**

**case** FIND\_COMPETITORS\_YET\_TO\_START**:**

Find\_Them**(** Num\_Entrants**,** this\_entrant**);**

**break;**

**case** FIND\_NUMBER\_ON\_COURSE**:**

Find\_Now**(**Num\_Entrants**,** this\_entrant**);**

**break;**

**case** FIND\_HOW\_MANY\_FINISHED**:**

Find\_Finished**(**Num\_Entrants**,** this\_entrant**);**

**break;**

**case** SUPPLY\_TIMES**:**

printf**(**"what is the competitor's id?\n"**);**

int guy **=** 0**;**

scanf**(** " %d"**,** guy **);**

printf**(**"what node have they reached?\n"**);**

int place **=**0**;**

scanf**(** " %d"**,** place**);**

this\_entrant**[**guy**].**location **=** place**;**

this\_entrant**[**guy**].**stat **=** 2**;**

printf**(**"what time did they reach it?\n"**);**

int hour**=**0**;**

int minutes**=**0**;**

scanf**(** " %d:%d"**,** hour**,** minutes**);**

**if(** this\_course**[**this\_entrant**[**guy**].**course**].**nodes**[**1**]** **==** place **)**

**{**/\*if this is the first place on their course then it sets the

\* start time

\*/

this\_entrant**[**guy**].**hours\_start **=** hour**;**

this\_entrant**[**guy**].**minutes\_start **=** minutes**;**

/\*else its adds to the time they have been racing\*/

**}** **else**

this\_entrant**[**guy**].**hours\_taken **=** this\_entrant**[**guy**].**hours\_start **-**

hour**;**

this\_entrant**[**guy**].**minutes\_taken **=** this\_entrant**[**guy**].**minutes\_start **-**

minutes**;**

**break;**

**case** READ\_IN\_FILE**:**

printf**(**"what is name of the file?\n"**);**

char file**[**50**];**

scanf**(** " %s"**,** file **);**

FILE **\*** new\_file**;**

new\_file **=** fopen**(**file**,** "r"**);**

/\*i have skipped over the first character as it is always the same for this\*/

fscanf**(**new\_file**,** " %d % d % d %d"**,** guy**,** this\_entrant**[**guy**].**location**,** hour**,** minutes**);**

this\_entrant**[**guy**].**stat **=** 2**;** //they are in the middle of the race

**if(** this\_course**[**this\_entrant**[**guy**].**course**].**nodes**[**1**]** **==** this\_entrant**[**guy**].**location **)**

**{**

this\_entrant**[**guy**].**hours\_start **=** hour**;**

this\_entrant**[**guy**].**minutes\_start **=** minutes**;**

**}** **else{**

this\_entrant**[**guy**].**hours\_taken **=** this\_entrant**[**guy**].**hours\_start **-**

hour**;**

this\_entrant**[**guy**].**minutes\_taken **=** this\_entrant**[**guy**].**minutes\_start **-**

minutes**;**

**}**

/\*if the place the racer is currently at is the node

\*occupied by the last position in the course array of nodes

\* then they have reached the end of their course \*/

**if** **(** this\_course**[**this\_entrant**[**guy**].**course**].**nodes**[**this\_course**[**this\_entrant**[**guy**].**course**].**number\_of\_nodes**]**

**==** place **)** this\_entrant**[**guy**].**stat **=** 3**;** //they have finished

**break;**

**case** PRODUCE\_RESULTS**:**

printf**(**"+++++++THE RESULTS+++++++"

"This is %s on the %s"

**,** this\_event**.**event\_name**,** this\_event**.**date**);**

int r **=**0**;**

**for** **(**r**=**1**;** r**<**Num\_Entrants**;** r**++)**

which **=** this\_entrant**[**r**].**stat**;**

**switch(**which**)** **{**

**case** 1**:**

printf**(**"%s has yet to start on course %c"**,** this\_entrant**[**r**].**name**,**

this\_entrant**[**r**].**course**);**

**break;**

**case** 2**:**

printf**(**"%s is at node %d on course %c and has taken %d hours and %d minutes"**,**

this\_entrant**[**r**].**name**,** this\_entrant**[**r**].**location**,** this\_entrant**[**r**].**course**,**

this\_entrant**[**r**].**hours\_taken**,** this\_entrant**[**r**].**minutes\_taken**);**

**break;**

**case** 3**:**

printf**(**"%s has finished course %c in %d hours and %d minutes "**,**

this\_entrant**[**r**].**name**,** this\_entrant**[**r**].**course**,** this\_entrant**[**r**].**hours\_taken**,**

this\_entrant**[**r**].**minutes\_taken**);**

**break;**

**}**

//break;

**case** QUIT**:**

printf**(**"you choose to quit"**);**

**break;**

**default:** printf**(**"that was not an option"**);**

**break;**

**}** **while** **(**command **!=** QUIT**);**

**return** **(**EXIT\_SUCCESS**);**

**}**

## Structures.h

/\*

\* File: structures.h

\* Author: James Upshall (jau1)

\*

\*

\*/

#ifndef STRUCTURES\_H

#define STRUCTURES\_H

#ifdef \_\_cplusplus

extern "C" **{**

#endif

struct event **{**

char event\_name**[**80**];**

char date**[**30**];**

char start\_time**[**5**];**

**};**

struct node**{**

int id**;**

char type**[**3**];**

int Connected\_Nodes**[**5**];**

**};**

struct track **{**

int id**;**

int start**;**

int end**;**

int time\_to\_complete**;**

**};**

struct course **{**

char id**;**

int number\_of\_nodes**;**

int nodes**[**20**];**

struct track **\***tracks**[**20**];**

**};**

struct entrant **{**

int id**;**

char name**[**51**];**

int location**;**

char status**[**51**];**

int stat**;**

int hours\_start**;**

int minutes\_start**;**

int hours\_taken**;**

int minutes\_taken**;**

char course**;**

**};**

void Find\_Him**(**char competitor**[],** int Num\_Entrants**,** struct entrant this\_entrant**[]);**

void Find\_Them**(**int Num\_Entrants**,** struct entrant this\_entrant**[]);**

void Find\_Now**(**int Num\_Entrants**,** struct entrant this\_entrant**[]);**

void Find\_Finished**(**int Num\_Entrants**,** struct entrant this\_entrant**[]);**

int read\_in\_event**(**char **\*** name\_filename**,** struct event this\_event**);**

int read\_in\_nodes**(**char **\*** name\_filename**,** struct node this\_node**[]);**

int read\_in\_tracks**(**char **\*** name\_filename**,** struct track this\_track**[],** struct

node **\*** this\_node**[]);**

int read\_in\_courses**(**char **\*** name\_filename**,** struct course this\_course**[]);**

int read\_in\_entrants**(**char **\*** name\_filename**,** struct entrant this\_entrant**[],**

int **\*** Num\_Entrants**);**

/\*Menu choices\*/

#define FIND\_COMPETITOR 1

#define FIND\_COMPETITORS\_YET\_TO\_START 2

#define FIND\_NUMBER\_ON\_COURSE 3

#define FIND\_HOW\_MANY\_FINISHED 4

#define SUPPLY\_TIMES 5

#define READ\_IN\_FILE 6

#define PRODUCE\_RESULTS 7

#define QUIT 8

#ifdef \_\_cplusplus

**}**

#endif

#endif /\* STRUCTURES\_H \*/

## RaceDataHandler.c

#include <stdio.h>

#include "structures.h"

/\*

\* File: RaceDataHandler

\* Author: James Upshall (jau1)

\*

\* This file contains all the functions to add data to the structures from

\* the data files

\*/

/\*method to read in the name file with the event data and put it

into the even struct\*/

int read\_in\_event**(**char **\*** name\_filename**,** struct event this\_event**)**

**{**

int success**=**1**;**

FILE **\*** name\_file**;**

name\_file **=** fopen**(**name\_filename**,** "r"**);**

**if** **(**name\_file **!=** **NULL)** **{**

fscanf**(**name\_file**,** "%[^\n] % [^\n] % [^\n]"**,** this\_event**.**event\_name**,**

this\_event**.**date**,** this\_event**.**start\_time**);**

**}** **else** **{**

printf**(**"Failed to open file\n"**);**

success **=** 0**;**

**}**

fclose**(**name\_file**);**

**return** success**;**

**}**

/\*method to read in the node file and to put it into the node struct\*/

int read\_in\_nodes**(**char **\*** name\_filename**,** struct node this\_node**[])** **{**

int success**=**1**;**

FILE **\*** name\_file**;**

int status **=**1**;**

name\_file **=** fopen**(**name\_filename**,** "r"**);**

**if** **(**name\_file **!=** **NULL)** **{**

int n **=** 0**;**

**for(**n **=** 1**;** n**<**30**;** n**++)** **{** //starting at 1 so that it is same as ids

**if(**status **!=** EOF**){** /\*moving through the data file until the end \*/

status **=** fscanf**(**name\_file**,** " %d"**,** " %s"**,** this\_node**[**n**].**id**,**

this\_node**[**n**].**type**);**/\*adding the values to the node structures \*/

**}**

**}**

**}** **else** **{**

printf**(**"Failed to open file\n"**);**

success **=** 0**;**

**}**

fclose**(**name\_file**);**

**return** success**;**

**}**

int read\_in\_tracks**(**char **\*** name\_filename**,** struct track this\_track**[],** struct

node **\*** this\_node**[])** **{**

int success**=**1**;**

FILE **\*** name\_file**;**

int status **=** 1**;**

struct node **\***st\_ptr1**;**

struct node **\***st\_ptr2**;**

name\_file **=** fopen**(**name\_filename**,** "r"**);**

**if** **(**name\_file **!=** **NULL)** **{**

**if(**status**!=** EOF**){**

int n **=** 1**;**

**for(**n**=**1**;** n**<** 50 **&&** status **!=** EOF**;** n**++){**

status **=** fscanf**(**name\_file**,** " %d"**,** " %d" " %d" " %d"**,** this\_track**[**n**].**id**,**

this\_track**[**n**].**start**,** this\_track**[**n**].**end**,** this\_track**[**n**].**time\_to\_complete**);**

/\*this is a very messy way to add the nodes that a node

leeds onto to the array of connected nodes\*/

//if(st\_ptr1->Connected\_Nodes[1] == 0){

st\_ptr1**->**Connected\_Nodes**[**1**]** **=** st\_ptr2**->**id**;**

// } else { st\_ptr1->Connected\_Nodes[2] = st\_ptr2->id;

// }

**}**

**}**

**}**

**else** **{**

printf**(**"Failed to open file\n"**);**

success **=** 0**;**

**}**

fclose**(**name\_file**);**

**return** success**;**

**}**

int read\_in\_courses**(**char **\*** name\_filename**,** struct course this\_course**[])** **{**

int success**=**1**;**

FILE **\*** name\_file**;**

int status **=** 1**;**

name\_file **=** fopen**(**name\_filename**,** "r"**);**

**if** **(**name\_file **!=** **NULL)** **{**

**if(**status**!=** EOF**){**

int n **=** 0**;**

**for** **(**n **=** 1**;** n **<** 5**;** n **++){**

status **=** fscanf**(**name\_file**,** "%c"**,** " %d" **,** this\_course**[**n**].**id**,**

this\_course**[**n**].**number\_of\_nodes**);**

int num\_nodes **=** 0**;**

**for(**num\_nodes **=** 1**;** num\_nodes **<=** this\_course**[**n**].**number\_of\_nodes**;**

num\_nodes**++){**

fscanf**(**name\_file**,** " d"**,** this\_course**[**n**].**nodes**[**num\_nodes**]);**

**}**

**}**

**}** **else** **{**printf**(**"Failed to open file\n"**);**

success **=** 0**;**

**}**

fclose**(**name\_file**);**

**return** success**;**

**}** **}**

int read\_in\_entrants**(**char **\*** name\_filename**,** struct entrant this\_entrant**[],**

int **\*** Num\_Entrants**)** **{**

int success**=**1**;**

FILE **\*** name\_file**;**

int status **=**1**;**

name\_file **=** fopen**(**name\_filename**,** "r"**);**

**if** **(**name\_file **!=** **NULL)** **{**

**}**

**if(**status **!=** EOF**){**

int n **=** 0**;**

**for(** n **=** 1**;** n **<** 30 **&&** status **!=** EOF**;** n**++){**

/\*moving through the data file until the end \*/

status **=** fscanf**(**name\_file**,** " %d"**,** " %c" "% [^\n]" **,** this\_entrant**[**n**].**id**,**

this\_entrant**[**n**].**course**,** this\_entrant**[**n**].**name**);**

/\*adding the values to the entrant structures \*/

this\_entrant**[**n**].**stat **=** 1**;**

Num\_Entrants **=** n**;** /\* storing away the number of entrants for use later\*/

**}**

**}**

**else** **{**

printf**(**"Failed to open file\n"**);**

success **=** 0**;**

**}**

fclose**(**name\_file**);**

**return** success**;**

**}**

## NowThisIsPodRacing

/\*

\* File: NowThisIsPodRacing.c

\* Author: James Upshall (jau1)

\*

\* This file is were I have put all the the functions which manipulate

\* the data depending on what the user wants to do

\*/

#include <stdio.h>

#include "structures.h"

void Find\_Him**(**char competitor**[],** int Num\_Entrants**,** struct entrant this\_entrant**[])** **{**

int i **=** 0**;**

int found **=** 0**;**

**for(**i **=** 0**;** i**<** Num\_Entrants**;** i**++){**

**if(**competitor **==** this\_entrant**[**i**].**name**)** **{**

printf**(**"%s was found, they are %s \n"**,** competitor**,** this\_entrant**[**i**].**status**);**

found **=** 1**;**

**}**

**}**

**if** **(**found **==** 0 **){**

printf**(**"%s could not be found"**,**competitor**);**

**}**

**}**

/\*these three functions are very similar as the just iterate through

the structures and count the incidents\*/

void Find\_Them**(**int Num\_Entrants**,** struct entrant this\_entrant**[])** **{**

int i **=** 0**;**

int NS **=**0**;**

**for(**i **=** 0**;** i**<** Num\_Entrants**;** i**++){**

**if(**this\_entrant**[**i**].**stat **==** 1**)** **{**

NS**++;**

**}**

**}**

printf**(**"there are %d competitor(s) whom have not yet started"**,** NS**);**

**}**

void Find\_Now**(**int Num\_Entrants**,** struct entrant this\_entrant**[]){**

int i **=** 0**;**

int ON **=**0**;**

**for(**i **=** 0**;** i**<** Num\_Entrants**;** i**++){**

**if(**this\_entrant**[**i**].**stat **==** 2**)** **{**

ON**++;**

**}**

**}**printf**(**"there are %d competitor(s) are on the course"**,** ON**);**

**}**

void Find\_Finished**(**int Num\_Entrants**,** struct entrant this\_entrant**[]){**

int i **=** 0**;**

int FIN **=**0**;**

**for(**i **=** 0**;** i**<** Num\_Entrants**;** i**++){**

**if(**this\_entrant**[**i**].**stat **==** 3**)** **{**

FIN**++;**

**}**

**}**printf**(**"there are %d competitor(s) finished"**,** FIN**);**

**}**

# Screenshot Tests