

16.216: ECE Application Programming

Spring 2013

Programming Assignment #11: Working with File I/O and Structures

Due **Wednesday, 5/1/13**, 11:59:59 PM

1. Introduction

This assignment will give you practice working with file input and structures. You will read and display a list of movies, including their ratings, running times, and scheduled start times, using commands to determine the exact output. This program will test your abilities to read input from a file and to organize data using structures.

2. Deliverables

Submit your source file directly to Dr. Geiger (Michael.Geiger@uml.edu) as an e-mail attachment. Ensure your source file name is ***prog11_movies.c***. You should submit only the .c file. Failure to meet this specification will reduce your grade, as described in the program grading guidelines.

3. Specifications

Input: At the start of each program run, you should prompt for the name of a file that holds the list of movies and their relevant information. If the file cannot be opened, your program should print an error and repeat the prompt until a valid file name is entered. **You must design a structure to hold the information about each movie; failure to do so will result in a 20 point deduction.**

The input file has the following characteristics:

- There will be at most 10 movies, although the exact number of movies is not listed in the file. ($1 \leq (\# \text{ of movies}) \leq 10$)
 - The first theater will always be theater number 1.
- The information about each movie is organized as follows:
 - Line 1: Theater number ($1 \leq (\text{theater number}) \leq 10$)
 - Line 2: Movie name
 - Movie names will be at most 40 characters.
 - Line 3: Movie rating (i.e., G, PG, PG-13, or R)
 - Line 4: Length of movie (in minutes)
 - Lines 5-8: Four starting times for movie.
 - All times use the format **h:mm** or **hh:mm**.

Two sample input files are posted on the course web page. Note that, in order to get your program to read these input files correctly, they should be placed in the same directory as your source file.

Input (cont.) and Output: After reading the input file, your program should handle each of the following commands:

- `all`: Print a list of all movies
 - The first line printed should be a set of headings as follows: "Theater", "Movie (Rating)".
 - All lines that follow should be formatted as shown below. Note that theater numbers should be printed so that they line up as shown:

```
Theater  Movie (Rating)
  1      Blue Valentine (R)
  2      The Company Men (R)
  3      The Eagle (PG-13)
  4      Gnomeo and Juliet 3D (G)
  5      The Green Hornet 3D (PG-13)
  6      Just Go With It (PG-13)
  7      The King's Speech (R)
  8      No Strings Attached (R)
```

- `number`: Print all information about a single movie in a particular theater.
 - After entering this command, the program should prompt the user to enter a theater number. Invalid numbers should trigger an error and cause the program to repeat its prompt for a theater number.
 - Given a correct theater number, information about the movie should be printed as shown below:

```
The Company Men (R)
113 minutes
1:45  4:30  6:55  9:30
```

- `name`: Print all information about a single movie with a matching name.
 - After entering this command, the program should prompt the user to enter the name of a movie. If no movie by that name is listed, the program should print an error and repeat its prompt for a movie name.
 - Given a correct movie name, information about the movie should be printed as shown in the example for the `number` command.
- `exit`: End the program.

If the user enters any other command, print an error message.

Output: All output specifications are given in the section above.

Error checking: As specified above, your program must check for the following errors:

- Unable to open input file
- Invalid commands
- Invalid theater numbers (for the `number` command)
- Invalid movie names (for the `movie` command)

4. Hints

Design process: This assignment is best completed by starting with a small problem and building up your solution piece by piece. I suggest doing the following:

1. Start with your file input. Create a file holding information about a single movie and read that information into appropriate variables.
 - To check your code's correctness, I suggest using the debugger to step through the program and see each variable change as it is read.
2. Expand your code to read information about all movies in the file; once again, check its correctness using the debugger.
3. Now, start handling user input. Start by implementing the `exit` command so that you can easily exit the program.
4. Write code to handle the `all` command, printing each theater number and the name and rating of the movie playing when that command is input.
5. Write code to handle the `number` command, reading the theater number and then printing all information about the corresponding movie, using the format described above, when that command is input.
6. Write code to handle the `name` command.

Removing newlines: Depending on how you read your inputs, you may have a newline character at the end of the input string—remember that the `fgets()` function reads an entire line, including the newline character.

The easiest way to handle this task is to explicitly overwrite the newline with a null terminator (`'\0'`). Say you have a character array, `str[]`, that contains a string with six characters, including a newline. To overwrite the sixth character, do the following:

```
str[5] = '\0';
```

Note: you do have to account for a single case in every program run in which blindly replacing the last character of a line with a null terminator will yield an incorrect result.

Functions: The one function that may be useful for this program—although you may write others—is a function that prints all information about a single movie using the formatting above, given that you have to perform that operation in two different places (the `name` and `number` commands). Remember, structures passed as function arguments are most efficiently passed by address.

5. Test Cases

Your output should match these test cases exactly for the given input values. I will use these test cases in grading of your assignment, but will also generate additional cases that will not be publicly available. Note that these test cases may not cover all possible program outcomes. You should create your own tests to help debug your code and ensure proper operation for all possible inputs.

```
C:\Windows\system32\cmd.exe
Enter schedule file name: schedule.txt
Enter command: all
Theater  Movie <Rating>
 1      Oblivion <PG-13>
 2      42 <PG-13>
 3      The Croods <PG>
 4      Evil Dead <2013> <R>
 5      Oz The Great and Powerful 3D <PG>
 6      Scary Movie U <PG-13>
 7      Olympus Has Fallen <R>
 8      G.I. Joe: Retaliation 3D <PG-13>
 9      Jurassic Park 3D <PG-13>
10     The Place Beyond the Pines <R>

Enter command: number
Enter theater number: 3
The Croods <PG>
91 minutes
12:10 2:30 5:00 7:30

Enter command: number
Enter theater number: 11
Error: invalid theater number 11
Enter theater number: 8
G.I. Joe: Retaliation 3D <PG-13>
99 minutes
12:30 3:05 5:40 8:15

Enter command: name
Enter movie name: 42
42 <PG-13>
128 minutes
12:10 3:10 6:10 9:10

Enter command: name
Enter movie name: Evil Dead
Error: invalid movie name Evil Dead
Enter movie name: Evil Dead <2013>
Evil Dead <2013> <R>
91 minutes
2:20 4:40 6:55 9:15

Enter command: quit
Invalid command quit

Enter command: exit
Press any key to continue . . .
```