16.216: ECE Application Programming

Fall 2012

Programming Assignment #9: Working with File I/O and Structures

Due **Wednesday, 12/5/12**, 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 ***prog9_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 read a file, called **`schedule.txt`**, that holds the list of movies and their relevant information. If the file cannot be opened, your program should print an error and exit. **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 ≤ (# of movies) ≤ 10)
    - The first theater will always be theater number 1.
- The information about each movie is organized as follows:
    - Line 1: Theater number (1 ≤ (theater number) ≤ 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**.

A sample input file is posted on the course web page. Note that, in order to get your program to read this input file correctly, it 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
    - o The first line printed should be a set of headings as follows: "Theater", "Movie (Rating)".
    - o 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.
    - o 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.
    - o 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.
    - o 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.
    - o 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 `schedule.txt`
- 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.

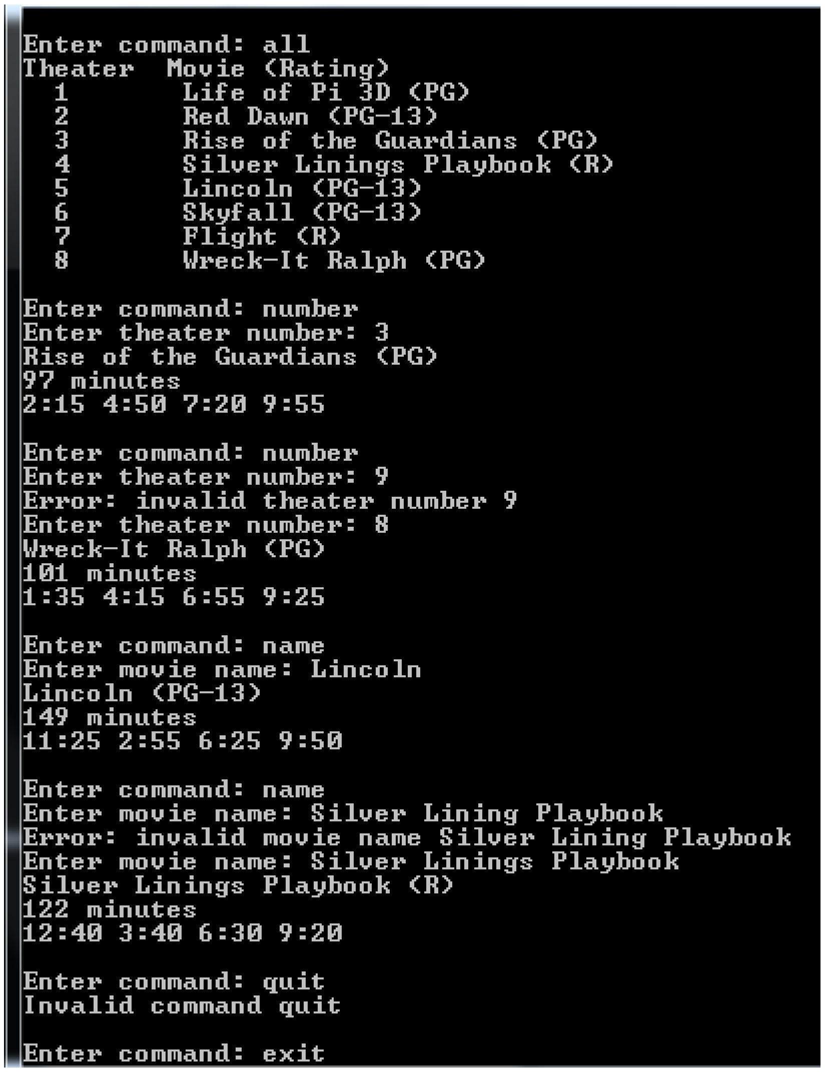
The easiest way to handle this task is to explicitly overwrite the newline with a null terminator (`'\0'`). Say you have a character array, `str[]`, that contains a string with six characters, including a newline. To overwrite the sixth character, do the following:

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

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 lab, 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.



```
Enter command: all
Theater  Movie (Rating)
  1       Life of Pi 3D (PG)
  2       Red Dawn (PG-13)
  3       Rise of the Guardians (PG)
  4       Silver Linings Playbook (R)
  5       Lincoln (PG-13)
  6       Skyfall (PG-13)
  7       Flight (R)
  8       Wreck-It Ralph (PG)

Enter command: number
Enter theater number: 3
Rise of the Guardians (PG)
97 minutes
2:15 4:50 7:20 9:55

Enter command: number
Enter theater number: 9
Error: invalid theater number 9
Enter theater number: 8
Wreck-It Ralph (PG)
101 minutes
1:35 4:15 6:55 9:25

Enter command: name
Enter movie name: Lincoln
Lincoln (PG-13)
149 minutes
11:25 2:55 6:25 9:50

Enter command: name
Enter movie name: Silver Lining Playbook
Error: invalid movie name Silver Lining Playbook
Enter movie name: Silver Linings Playbook
Silver Linings Playbook (R)
122 minutes
12:40 3:40 6:30 9:20

Enter command: quit
Invalid command quit

Enter command: exit
```

Remember, to get your program to terminate with a message saying, “Press any key to continue …”, use the **Start Without Debugging** command (press Ctrl + F5) in Visual Studio to run your code.