Assignment 1: Movies A↓

Due Oct 10, 2022 by 11:59pm   **Points** 95   **Submitting** a file upload
File Types zip   **Available** Sep 21, 2022 at 12:01am - Oct 12, 2022 at 11:59pm

---

This assignment was locked Oct 12, 2022 at 11:59pm.

Introduction

In this assignment, you'll write a program that will introduce you to programming in C on UNIX based systems.

Learning Outcomes

- How to write a C program to solve a problem? (Module 1 MLO 4)
- How do you interact with the user in C programs? (Module 1 MLO 6)
- How are C programs transformed into an executable form? (Module 1 MLO 7)
- Define programming language constructs in C (Module 2 MLO 1)
- Describe structure data types of C programming language (Module 2 MLO 3)
- What are pointers and how is this mechanism supported in C? ((Module 2 MLO 4)
- Describe memory allocation in Unix (Module 2 MLO 5)
- Explain how to create and manipulate strings in C (Module 2 MLO 6)

Instructions

Write a program that

- Reads a CSV file with movie data that is provided to the program as an argument
- Processes the data in the file to create structs to hold data for each movie
- Creates a linked list of all these structs
- Gives user choices to ask questions about the movies in the data
- Prints out the data about the movies per user choice

Format of the CSV File

Here is a **sample file (https://canvas.oregonstate.edu/courses/1890464/files/94288613/download)** whose format corresponds to the format of the CSV file your program will be tested with.

- The first line in the input file will always be the header line containing the column headers (i.e., it will not have data for a movie).
- All the other lines will have the data for a movie in valid format and no columns will be missing.
- The file will not have any empty lines.
- The file will contain data for at least one movie.
- Don't assume a maximum number of movies in the file.
- Commas appear as delimiters between columns, but will not appear in the values of any columns.

- The name of the input file will be less than 50 characters and will not contain any spaces.
- Don't assume that the input file name will have a particular extension.

This file has the following columns:

1. Title
   - This is a string with the movie title.
   - E.g., Iron Man 2
   - You cannot assume a maximum length of the movie title.
2. Year
   - This is a 4 digit integer value for the year the movie was released
   - It's value will be between 1900 and 2021 (inclusive of these years).
   - E.g., 2010
3. Languages
   - The language or languages in which the movie was released.
   - One or more string values that are always enclosed within []
   - Multiple values are separated by semi-colons.
   - E.g,
     - [English;Portuguese;Spanish]
     - [English;French]
     - [English]
   - You can assume that the maximum number of languages any movie can be released in is 5.
   - You can assume that the maximum length of a particular language is 20 characters.
   - You cannot assume any particular case for the letters in the language.
     - E.g., don't assume that the first letter is upper case, or that all letters are lowercase.
4. Rating Value
   - A number between 1 and 10 (inclusive of both 1 and 10)
   - It can be an integer or a double with one digit after the decimal point
   - E.g.,
     - 5
     - 8.7

Program Functionality

Process the input file

When your program starts it must read all data from the file and process it. After processing the file, the program must print the following message to stdout

"Processed file XYZ and parsed data for M movies"

where XYZ is the name of the file that has been process and M is the number of movies whose data has been processed (this will be 1 less than than the number of lines in the file because the first line has the column headers).

E.g., Processed file movies_sample_1.csv and parsed data for 24 movies

Interactive Functionality

Next your program should display a menus of interactive choices to the user and process the selected choice as described below. For the text of messages to print, see the section "Sample Program Execution."

1. Show movies released in a specified year

- If the user chooses this option, then ask them to enter a year and
  - Display the names of all the movies released in that year, one on each line
- If the data doesn't have any movies released in that year, print a message about this.
- Your program can assume that the user will enter a 4 digit integer for the year between 1900 and 2021 (inclusive of these years)

2. Show highest rated movie for each year

- If the user chooses this option, then for each year for which at least one movie was released, display a movie that had the highest rating along with the year and the rating with one line per year.
  - In case of ties, display any one movie that had the highest rating that year.
- Display the data in the form: YYYY RatingValue MovieTitle
- The data doesn't have to be sorted by year or by rating value.
- E.g.,
  - 2010 8.5 Avengers: Infinity War
  - 2012 8.1 The Avengers

3. Show movies and their year of release for a specific language

- If the user chooses this option, ask them to enter a language and
  - For all movies released in the specified language
  - Display the year of release and the movie title, one line per movie
- If the data doesn't include any movie released in this language, print a message about it.
- You should only do an exact match on the language entered by the user
  - E.g., "English" shouldn't match "english"
- You can assume that the length of the language string entered by the user will be less than 20 character.

4. Exit

- If the user choose this option, the program should exit.

Notes:

- For the interaction choice if the user enters an incorrect integer (i.e., something other than 1 to 4), print an error message and again present the 4 choices to the user.
- You can assume that when the program asks user to enter an integer, the user will indeed enter an integer (i.e., you don't need to verify the data type of the user input).
- Don't assume that the languages in the test file will be the same as the ones that appear in the sample file.

- Don't make an assumption about the number of distinct languages that can appear in the test file.

Technical Requirements

Your program is required to

1. Read data from the file line by line
2. Break-up the line into tokens
3. Create a struct called `movie` with the data for that line
4. Create a linked list containing all the movie structures

See the Resources section (below) to access sample code that you can adapt to meet these technical requirements.

Sample Program Execution

Here is a complete example of executing the assignment with the provided sample CSV file.

```
$ ./movies movies_sample_1.csv
Processed file movies_sample_1.csv and parsed data for 24 movies

1. Show movies released in the specified year
2. Show highest rated movie for each year
3. Show the title and year of release of all movies in a specific language
4. Exit from the program

Enter a choice from 1 to 4: 1
Enter the year for which you want to see movies: 1999
No data about movies released in the year 1999

1. Show movies released in the specified year
2. Show highest rated movie for each year
3. Show the title and year of release of all movies in a specific language
4. Exit from the program

Enter a choice from 1 to 4: 1
Enter the year for which you want to see movies: 2012
The Avengers
Rise of the Guardians
Anna Karenina

1. Show movies released in the specified year
2. Show highest rated movie for each year
3. Show the title and year of release of all movies in a specific language
4. Exit from the program

Enter a choice from 1 to 4: 2
2008 7.9 Iron Man
2009 7.6 Sherlock Holmes
2010 7.0 Iron Man 2
2013 7.2 Iron Man 3
2017 7.9 Thor: Ragnarok
2012 8.1 The Avengers
2016 7.8 Captain America: Civil War
2018 8.5 Avengers: Infinity War
2015 7.4 Avengers: Age of Ultron
2011 7.0 Thor
2014 7.8 Captain America: The Winter Soldier
2003 6.6 Right on Track

1. Show movies released in the specified year
2. Show highest rated movie for each year
3. Show the title and year of release of all movies in a specific language
4. Exit from the program
```

```
Enter a choice from 1 to 4: 3
Enter the language for which you want to see movies: English
2008 The Incredible Hulk
2009 Sherlock Holmes
2008 Iron Man
2010 Iron Man 2
2013 Iron Man 3
2017 Thor: Ragnarok
2012 The Avengers
2016 Doctor Strange
2018 Avengers: Infinity War
2015 Avengers: Age of Ultron
2011 Thor
2013 Thor: The Dark World
2017 Spider-Man: Homecoming
2011 Captain America: The First Avenger
2016 Captain America: Civil War
2015 Ant-Man
2014 Captain America: The Winter Soldier
2018 Mary Queen of Scots
2016 Revolting Rhymes Part One
2017 The Glass Castle
2016 Free Fire
2003 Right on Track
2012 Rise of the Guardians
2012 Anna Karenina

1. Show movies released in the specified year
2. Show highest rated movie for each year
3. Show the title and year of release of all movies in a specific language
4. Exit from the program

Enter a choice from 1 to 4: 3
Enter the language for which you want to see movies: Punjabi
No data about movies released in Punjabi

1. Show movies released in the specified year
2. Show highest rated movie for each year
3. Show the title and year of release of all movies in a specific language
4. Exit from the program

Enter a choice from 1 to 4: 5
You entered an incorrect choice. Try again.

1. Show movies released in the specified year
2. Show highest rated movie for each year
3. Show the title and year of release of all movies in a specific language
4. Exit from the program

Enter a choice from 1 to 4: 4
$
```

Hints

- Code the functionality incrementally, constantly testing it and using version control (e.g., Git) to track changes.
  - You can use the Grading Rubric as a guide to incrementally develop the functionality.
- To convert a string to int, you can use the function **atoi** ⬗ **(https://man7.org/linux/man-pages/man3/atoi.3.html)** .
- To convert a string to a double, you can use the function **strtod** ⬗ **(https://man7.org/linux/man-pages/man3/strtod.3.html)** .
- To extract tokens from a string, use **strtok_r** ⬗ **(https://man7.org/linux/man-pages/man3/strtok_r.3.html)** instead of strtok.

- To print a variable of data type double to one decimal space, use "%.1f" for this variable in the format string passed to **printf** ⤷ **(https://man7.org/linux/man-pages/man1/printf.1.html)**.

Resources

- We have provided an example program **main.c** ⤷ **(https://repl.it/@cs344/studentsc#main.c)** that reads a space delimited text file with student data, processes it line by line, creates a linked list of student structs with that data and then prints that data.
  - In your assignment, you can use and adapt any code from this program and also any code presented in the explorations.
  - A sample file with student info that you can use with this program is available **here** ⤷ **(https://repl.it/@cs344/studentsc#student_info1.txt)**.
  - You can compile and run the program by using the following commands:

```
gcc --std=gnu99 -o students main.c
./students student_info1.txt
```

What to turn in?

- You can only use C for coding this assignment and you must use the gcc compiler.
- You can use C99 or GNU99 standard or the default standard used by the gcc installation on os1.
- Your assignment will be graded on os1.
- Submit a single zip file with all your code, which can be in as many different files as you want.
- This zip file must be named `youronid_program1.zip` where youronid should be replaced by your own ONID.
  - E.g., if chaudhrn was submitting the assignment, the file must be named `chaudhrn_program1.zip`.
- In the zip file, you must include a text file called `README.txt` that contains instructions on how to compile your code using gcc to create an executable file that must be named `movies`.
- When you resubmit a file in Canvas, Canvas can attach a suffix to the file, e.g., the file name may become `chaudhrn_program1-1.zip`. Don't worry about this name change as no points will be deducted because of this.

Grading Criteria

- The value of items that are being graded are described in the grading rubric.
- The grading will be done on os1 by running the program against one or more CSV files that have the exact format of the sample CSV file but with different data.

Assignment 1 Movies

Criteria	Ratings		Pts
Message about processed file and movies is correct	**15 pts** **Full Marks**	**0 pts** **No Marks**	15 pts
"Show movies released in the specified year" works correctly	**15 pts** **Full Marks**	**0 pts** **No Marks**	15 pts
"Show highest rated movie for each year" works correctly	**20 pts** **Full Marks**	**0 pts** **No Marks**	20 pts
"Show the title and year of release of all movies in a specific language" works correctly	**20 pts** **Full Marks**	**0 pts** **No Marks**	20 pts
The program exits only when user enters the correct choice	**4 pts** **Full Marks**	**0 pts** **No Marks**	4 pts
Top level choice other than 1 to 4 is correctly handled	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
The code is well-commented	**5 to >0.0 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
The program meets the technical requirement of processing the file line by line	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
The program meets the technical requirement of defining and using an appropriate movie struct for the data of a movie	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
The program meets the technical requirement of using a linked list for all the movie structs	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
		Total Points: 95	