CIS*2520 Data Structures Assignment 3

Due: Sunday, Nov. 3, 2019, 11:59pm
Version 1.01 (changes highlighted in yellow)

For this assignment you will be building a database based on IMDB and index the records using multiple binary trees.

GITIGNORE
Create a text file in your working directory called ".gitignore". Type the following lines into that file:

*.gz
*.tsv

This is a requirement for your assignment. If this file is not found in your git directory you will lose marks.

THE DATA
You will use data from https://datasets.imdbws.com and create arrays as follows:

1) Read name.basics.tsv once to determine the number of entries that have the word "actor" in the column "primaryProfession".
2) Allocate an array of
```
struct name_basics {
        char *nconst;
        char *primaryName;
};
```
For each entry in the file that matches the criterion in "1)".
3) Read the file again to store the relevant data into the structure defined in "2)". For each line in the file allocate the correct amount of memory for "nconst" and "primary_name".
4) Repeat steps 1-3 for the file title.basics.tsv, and the structure:
```
struct title_basics {
        char *tconst;
        char *primaryTitle;
};
```
For rows in the table where titleType=movie and isAdult=0.
5) Repeat steps 1-3 for the file title.principals.tsv, and the structure:
```
struct title_principals {
        char *tconst;
        char *nconst;
```

```
        char *characters;
    };
```

For rows in the table where category=actor.

Finding the files. I have uploaded the files to the linux.socs.uoguelph.ca server in the directory: /home/courses/cis2520.

You can also download your own copies from https://datasets.imdbws.com and decompress them using gunzip. WARNING: these are big files. Don't store these files in the same directory as you code (put them in a different directory).

THE TREES

Build 6 binary trees, by traversing the arrays as follows:

1) A tree based on the key nconst that references the structures in the name_basics array. (Since the nconst values are in ascending order in the input file, you will need to add items to the tree in a clever way so that you don't end up with a really long linked list).
2) A tree based on the key primaryName that references the structures in the name_basics array.
3) A tree based on the key tconst that references the structures in the title_basics_array.
4) A tree based on the key primaryTitle that references the structures in the title_basics_array.
5) A tree based on the key tconst that references the structures in the title_principals_array.
6) A tree based on the key nconst that references the structures in the title_principals_array.

THE PROGRAM

You will create a program, called a3, that uses the binary trees to perform two operations:

1) It will take the primaryName of an actor, look up its entries in the name_basics array and determine the corresponding nconst values. Then it will look up the nconst values in the title_principals array to determine the corresponding tconst values. Finally, it will look up the tconst values in the title_basics_array and it will print out the corresponding primaryTitle and characters.
2) It will take a primaryTitle and look up its entries in the title_basics_array to get the tconst value(s). It will then use the tconst value(s) in the title_principals array to retrieve the relevant nconst values. Finally, it use the nconst values to retrieve the primaryName values and print out the primaryName and characters.

Your program will print a prompt consisting of two characters, "> " (a greater than sign, followed by a space). If the user types "name" it will perform the first operation, and if the user types "title" it will perform the second operation. After the first word, the user must type either a primaryName exactly as it appears in the table, or a primaryTitle exactly as it appears in the table. Users are allow to include any number of spaces before the first word and any number of spaces after the first word. The primaryName and the primaryTitle must be spaced,

punctuated and spelled exactly as in the table. There can be any number of spaces at the end of the line.

The output will consist of the two values from the table, separated by three characters, " : " (a space, a colon, a space).

Your program will take a single command line argument with the name of the directory where the files are located. Your program should run correctly on the linux.socs.uoguelph.ca server when run as: "./a3 /home/courses/cis2520".

ORGANIZING YOUR CODE

Create .c and .h files with the following contents:

common – handles any common functions (e.g. to open and read files),
binary – handles the binary trees,
name – handles the name_basics array,
title – handles the title_basics array,
principals – handles the title_principals array

Create a main.c file that implements the user input and printed output. And, create a makefile that compiles your each of your .c files into a .o file using the -Wall -ansi -pedantic flags, and then creates an executable called a3.

SUBMITTING YOUR CODE

Submit your .c, .h and makefile via git.

Put your full name, student ID number, and uoguelph e-mail at the top of each file.

Use consistent indenting, formatting, commenting, naming and other good programming practises to make your code readable.