Programming Assignment 5, Fall 2017

OBJECTIVE

The objective of this assignment is to give you more practice with using structures and binary random access files.

ASSIGNMENT SUBMISSION

To get credit for this assignment, you must

- ✓ write a multi-file program in C
- ✓ submit your project as a tar file through Canvas exactly as instructed (naming, compatibility, etc.)
- ✓ submit your assignment on time

PROBLEM STATEMENT

For this assignment, you need to write a program that displays a text menu and, depending on the user input, manipulates the binary file called `animals.dat`. Valid input choices are 0-5, where 0 indicates a user wants to quit, while 1-5 indicate they want a particular operation to be performed. After each operation 1-5 is performed, the program is to continue, display the menu again, and react to the new user input.

Menu choices and processing logic

If the user enters 1, your program is to perform processing similar to the *strings* command in Unix. In other words, the program is to read in an entire binary file in a sequential fashion one byte at a time and for each byte, if the byte content is consistent with an ASCII value 32-127, the program is to write it out to console as an ASCII character. If the value is not consistent with a printable character, the program is to print a space character to the console. Each line in the terminal is to be no longer than 80 characters. After the text version of the binary file is written out to the console, the program is to go back to the initial menu.

If the user enters 2, your program is to perform processing similar to the way numbers are displayed in the *ghex* editor. In other words, the program is to read in an entire binary file in a sequential fashion, four bytes at a time, and for each four-byte unit, it is to write it out to console as a signed integer. When writing to console, make sure you also print a space between all integer values and that the line is no longer than about 80 characters. After the integer version of the binary file is written out to the console, the program is to go back to the initial menu.

If the user enters 3, you are to assume that the input file `animals.dat` consists of records of animal type (defined below) and you need to ask the user which animal record to find and display to the screen – indicated by an integer denoting the record number. The animal type structure is defined as:

```
#pragma pack(1)
struct animal {
    short int id;
    char name[20];
    char species[35];
    char size;
    short int age;
};
```

Since the structure memory layout is different on different machines, the pragma pack directive forces the compiler to store variables of animal type the same way (using the same alignment / padding) regardless of the platform. Note that this type of search is to use random file access and that the record number and the id of an animal are the same entity. For example, if the file consists of entries like:

```
1,Allegra,Pseudois nayaur,S,5
2,unknown,Ailurus fulgens,X,10
3,Athena,Moschus fuscus,X,2
```

then the 3rd record on that file is `3,Athena,Moschus fuscus,X,2`

All animals are listed in the increasing order by their id number, starting at value 1. If there is a hole in id numbers, e.g. 2, then the structure information is still present in the file, except the name component contains the string "unknown" to signify an empty record. Make sure your search uses random file processing.
If an invalid id is entered, in this example any value other than 1 or 3, your program is to display an error message. Otherwise, animal record is displayed. In either case, the program is to go back to the initial menu.

If the user enters 4, you are to assume that the input file `animals.dat` consists of records of animal type, as described above, and you need to ask the user which animal record to update – indicated by an integer denoting the id of the animal. A user can update any information they choose other than the animal id. Bear in mind that animal names and species could consist of more than one word. An update should make use of random file access and report an error if an invalid id is entered. If a valid id is used, an update is to be performed by asking the user to enter the data, and the program is to write this information back to the `animals.dat` file. After performing an update, the program is to go back to the initial menu.

If the user enters 5, you are to assume that the input file `animals.dat` consists of records of animal type, as described above, and you need to ask the user which animal record to overwrite – indicated by the record number. Assuming a valid value has been entered, the program is to overwrite the animal entry with the information from the last record in the file (other than the id) and then shrink the actual file so that it does not contain the last record - overall the logic is similar to the deletion from the array list we discussed in class. Look at function *truncate* or *ftruncate* to help you shrink the file. For example, if the file consists of entries like:

```
1,Allegra,Pseudois nayaur,S,5
2,Anjolie,Ailurus fulgens,X,10
3,Athena,Moschus fuscus,X,2
```

then if the user typed in 2, the resulting file would be:

```
1,Allegra,Pseudois nayaur,S,5
2,Athena,Moschus fuscus,X,2
```

Make sure you use random file access for this menu choice.
If an invalid id is entered, in this example any value outside of 1-3 range, your program is to display an error message. In either case, the program is to go back to the initial menu.

At the end of the program, when the user enters 0 to quit, write modified `animals.dat` contents to `animals.csv` file skipping all unknown animals (sample provided). Note that `animals.dat` itself, after the program run, should contain contents of the modified original file.

Program Specs

- The binary file is to be open only once per program run, i.e. do not keep reopening the same file for different value choices
- Do not assume that the exact same file will be used for grading your program, i.e. your program needs to be scalable and not dependent on the number of initial animals in the `animals.dat` file that is given to you for program development
- The only data types you are allowed to use in this program are primitive data types and struct animal. You are NOT allowed to use arrays or any other data structures
- Your program should use several functions (e.g. one function per menu choice)
- You are NOT allowed to use global variables
- You do NOT need to create an ADT for an animal but you should split the program into minimum 3 files, in which case .h should contain the structure definition and all function prototypes required by the program, .c should contain all function definitions, and driver.c that should contain main function only
- You need to create a makefile for your code that is to be called **makefile**
- Name the executable file `animalbin`
- Your program has to follow basic stylistic features, such as proper indentation (use whitespaces, not tabs), meaningful variable names, etc.
- You need to comment your code
    - Remember to include your name and a statement whether you tested your code on the cssgate server or Ubuntu 16.04 LTS Desktop 32-bit

- **Your program must compile in gcc gnu 90 – programs that do not compile will receive a grade of 0**
- All program files must reside in one folder and the contents of the entire folder must be compressed using tar utility (check lab 1 directions to make sure you are compressing an entire folder and not individual files); the folder is to be named **yournetid_pr5**
- Your tar files should be named **pr5.tar**

Extra Credit (15%)

Open-ended – extend the program in some fashion – explain in comments at the top of your code. For example, you may choose to add an animal (among other choices). Whatever operations you choose to encode, they must be consistent with the program/data logic, e.g. when you add an animal, you still need to keep the file sorted by the animal id and you should fill out any *unknown* entries first. Random file access processing ideas are worth more of an extra credit than the sequential ones and no helper data structures could be used.

Program Submission

On or before the due date, use the link posted in *Canvas* next to *Programming Assignment 5* to submit your tar file. Make sure you know how to do that before the due date