CSE 320 Spring 2019 HW #1

February 12, 2019

Deadline: February 24, 23:59 Stony Brook time (Eastern Time Zone).

1 Introduction

In this semester we are located in the Art center, which sets the theme of art for the assignments. This homework asks you to implement an in-memory database to store art pieces of an art collector. The application starts with some given budget and then the input file contains art pieces that art collector is going to buy. After the input file has been processed, the application should print the content of the in-memory database as well as remaining budget and worth of the art collection.

Goal of this homework is to get your hands dirty with programming in C language and become more familiar with the Linux environment. You need to implement an application that resembles a very simple in-memory database. It will help you to learn some of the C concepts, get familiar with C syntax, and will give you an idea of how programs in C are written.

2 Setting Up SSH Connection (Optional)

As you may have seen, using VMWare interface to work with CLI (Command Line Interface) of the VM may be not very convenient. There are several ways how to deal with it within VMWare but we are going to try a different approach. In the real world you may be working with remote servers even if you are front-end developers. In such case one of the most common solutions is to use so-called SSH (Secure Shell) connection.

You can find more about SSH in the online resources below
https://en.wikipedia.org/wiki/Secure_Shell
http://blog.robertelder.org/what-is-ssh

2.1 Obtaining IP Address

To connect to your VM you need its *IP address* and its *SSH connection port*. Default port for SSH is 22. To obtain its IP address, type the following in your

VM terminal:

```
$ ip a
```

You should see two network interfaces. One should have an IP address `127.0.0.1` and another one should have some other IP address. We need that second one, which is the IP address of your VM. Now we need an SSH client to connect to your VM. Please read the appropriate subsection depending on Operating System of your computer.

2.2 Linux OS

If you use Linux as your Operating System, most likely you already have the SSH client. Open the terminal (it is an application) and type the following:

```
$ ssh vm_login@vm_ip_address
```

where you need to use your VM login instead of **"vm_login"** and your VM IP address (that we found earlier) instead of **"vm_ip_adress"**.

2.3 macOS

If you use macOS as your Operating System, you already have the SSH client. Open the terminal (it is an application) and type the following:

```
$ ssh vm_login@vm_ip_address
```

where you need to use your VM login instead of **"vm_login"** and your VM IP address (that we found earlier instead) of **"vm_ip_adress"**.

I would recommend to learn and use an application called **iTerm2** instead of native terminal available in the Mac OS as it is rich in features. You can read more about it in the corresponding documentation https://www.iterm2.com.

2.4 Windows

If you are Windows user, you have several options but I will focus on two clients. One of the most used SSH clients for Windows is **PuTTY**. You can find instruction how to use it online under the following link:

PuTTY instructions

NB: You don't have to put PuTTY into "Windows" folder, you can put it anywhere you want. Usually, the best place for PuTTY is a Desktop.

PuTTY is good enough, however, I would recommend to try **SmarTTY**. It has several neat features and also simple to use. You can find more about it online under the following link:

http://smartty.sysprogs.com

2.5 Connection Refused

So we tried...

But we cannot connect as a remote host (your VM) keeps refusing access. It happens because by default machines do not expect that someone can log in into them. Let's fix it. We need to install an SSH server first, so type the following command in your VM's terminal:

```
$ sudo apt-get install openssh-server
```

If it will give you an error at the end, try perform update first and then repeat the command to install the SSH server (you most likely can ignore error from the update command):

```
$ sudo apt-get update
$ sudo apt-get install openssh-server
```

Now, the connection through SSH should work. Moreover, finally you can easily do copy and paste, have a much nicer screen resolution, and enjoy writing in C! It is worth mentioning that you can do even more, e.g., use SCP or features of a particular SSH client you may use. But that I recommend to learn on your own by consulting corresponding documentation.

3 Your Programming Task

In this homework you need to implement an application called `art_collector`. The application takes commands from a given file to modify internal database of art pieces and then return some result based on specified flags. You always start with a clean database (it is very inefficient but for the sake of this assignment it is okay) and then you start to modify you internal database implemented within your C code based on the content of the given file. Regarding flags you will have two scenarios:

- No flags provided. Then you need to print string **NO QUERY PROVIDED**.
- At least one flag is set. Then you need to print according to the flags.

Your program should work according to the following specification (being in square brackets means to be optional):

```
$ art_collector filename -b N [-v] [-i id]
                [-t type] [-n artist_name] [-o filename]
```

- -v Print information about all art pieces in the database.
- -i Print information about all art pieces with matching ID.
- -t Print information about all art pieces with matching TYPE.

- -n Print information about all art pieces with matching ARTIST NAME.
- -b Initial budget of N monies.
- -o Print information to the given file. If file does not exists then create one. If file exists, then prompt user if file should be overwritten. If user says "y" or "yes" then proceed. If user says "n" or "no" then exit and print message **FILE EXISTS**.

Result of your program should be written to `stdout` unless flag "-o" is set. Safe assumptions in this assignment are the following:

- You may assume that required arguments for flags are always provided.
- Commands (the very first word) in the input file are always in the upper case.

You program should comply to the following constraints:

- You cannot use string libraries, e.g. `<string.h>`.
- You cannot use `getopt()` function to parse flags.
- You cannot use indices for arrays. You have to use pointers and pointer arithmetic. It means that if you want to access the third element of the array `ptr`, you have to use `ptr + 2` instead of `ptr[2]`.
- Art type in the database should always be capitalized.
- Art name and Artist name in the database should always start with capital letter and the rest is lowercase.
- To store your art pieces records you will need to use some data structure. That data structure should be a linked list. It should be called `art_pieces`.
- You shall not modify the `.travis.yml` file.
- We should be able to compile your program using `make` command.

IMPORTANT: Failing to comply to any of the constraints may result into 100% penalty for the grade or test case.

In the real world you may receive some set of constraints or assumptions. However, it may be completely up to what to check and how to check to estimate how correct is your application. As this is a course where you are still learning, we provide you an idea of what you should check:

- Combination of flags is correct, e.g., "-v" flag cannot be combined with any other flag except "-o".
- Specified input file exists and can be opened.

- Command in the input file is correct.
- Input data makes sense.
- Art piece ID should be unique.

IMPORTANT: You should print only what is asked. Otherwise, you may file a test case and receive 0 points for it!

You may think that decorating your output or having just a minor difference such as printing a dot after printed text is a very minor thing. However, when you are asked to print a very particular result, it means that your output will be expected to be in a very specified form. If you don't follow the specification it is not others people code cannot handle yours and it is bad. It is your code is bad.

In general, it may be useful to use for your programming a combination of *Design by Contract* and *Defensive Programming*. You use first when it is about your output and you use latter when it comes to check all the inputs you have received. You can easily search online what those terms mean though we may cover that a bit during lectures.

3.1 Possible Commands in the Input File

There are three legit commands:

- BUY art_id art_type art_name artist_name price
- UPDATE art_id art_type art_name artist_name price
- SELL art_id price

3.2 Requirements to the Input Format

General rule is that everything that is not correct is incorrect.

There are several requirements for the input formats including flags:

- Art piece ID is an integer number greater than zero.
- Art type is one word of up to 10 letters in size.
- Art name can consist only of letters and spaces and its size is from 3 to 50 letters inclusive.
- Artist name can consist only of letters and spaces and its size is from 3 to 20 letters inclusive.
- Price is an integer number greater than zero.

3.3 Output Format

Output format for the art piece record is the following:

ID TYPE ART-NAME ARTIST-NAME PRICE

And they all separated by white space, for example,

15 PAINTING Girl on the ball Picasso 100

If you have more than one art piece record to print then just print each record on a new line. Format for each record should look like

`"%d %s %s %s %d"`

After all records you also should print available budget and worth of the art in the collection.

3.4 Errors/No Student Records to Print

As you shall not rely on neither user providing the correct combination of flags nor the input file being correct, you always should check for possible errors. In this assignment we provide list of some errors that you need to check with print statement that should be printed.

- **NO QUERY PROVIDED** if application executed without flags.
- **FILE EXISTS** see flag "-o" description.
- **FAILED TO PARSE FILE** if something wrong with one of the fields in the input file or there is an unknown command.
- **ID NOT UNIQUE** if trying to add a record with ID that already exists in the database.
- **RECORD CANNOT BE DELETED NOR UPDATED** if trying to delete a record that does not exist or trying to update a record by given ID that does not exist.
- **RECORD NOT FOUND** if flags were legit but there is no such record in the database.
- **NOT ENOUGH MONIES** if there is not enough monies in the budget to buy the next art piece.
- **OTHER ERROR** when there is an error but not within above categories.

3.5 Linked List

As stated in the assignment, you should use linked list to store are pieces records and later retrieve information about them. Moreover, elements in the list should be sorted in ascending order by art piece ID. It means that if element A of the list points to element B in the list (or you can say A goes before B in the list) then art piece ID stored in the A should be less than art piece ID stored in the B. It also implies that you cannot just append new records to the end of list. Sometimes you will need to insert them somewhere in the middle of the list.

3.6 Actual programming task

Now, as you are familiar with all the separate pieces of the application let's put all of them together. The user runs the application user following command:

```
$ art_collector filename -b N [-v] [-i id]
                [-t type] [-n artist_name] [-o filename]
```

The first parameter is the name of the input file and it is a mandatory parameter. It is safe to assume that it always comes as a first parameter. Another mandatory parameter is a budget. It indicates the initial amount of monies the art collector has, which later will be used to "buy" art pieces.

The next step is to parse the input file. Every time you encounter the "BUY" command you need to check if there is enough budget to purchase the art piece, add it to the database, and update the remaining budget accordingly. If it the "UPDATE" command then you need to update that particular art piece, e.g. the command may update the price that will affect art collection worth. Finally, if it "SELL" command then you need to remove art piece from the database and update the budget accordingly. Please note that you need to parse file line-by-line. It means that if the there is not enough monies to purchase the next art piece then you need to print corresponding error message and exit the program.

After file parsing is complete you need to print the records in database and print remaining budget and art collection worth, which is the sum or prices of every art piece in the database.

Important to note, that we will compile your program using `make` command. We provide useful material for the `Makefile` and `make` command in the readings sections.

3.7 Extra Credit

There are two possibilities for extra credit.

3.7.1 Maximizing art collection worth

In the homework you need to parse the input file line-by-line. It means that if you encounter the art piece with the price higher that currently available budget you should print the error message and exit the program. However, in this extra credit part of the assignment you need to ignore such fact and purchase all the art pieces from the database that will maximize the art collection worth.

3.7.2 Persistent database

As you may notice, every time we start the the application its internal memory database is empty. In this extra credit part of the assignment you need to create a persistent database. It means that the database should have records from the previous run. For example, if the database had records A, B, and C at the end

of the previous run then if we run your application again, these records should be in the database.

4 GitHub Repository and Code

Please use the same GitHub account that you have created for the HW#0. Once you are ready, please follow the link:

https://classroom.github.com/a/KxVqoQKX

which should create you a repository for this homework.

5 Useful Reading

In class we discussed that `main()` function can also take arguments. You can find a bit more information and see example under the following link:

http://crasseux.com/books/ctutorial/argc-and-argv.html

On top of that, we discussed that there is a better way to deal with flags. More specifically, using `getopt()` function. More information is available under the following link:

https://www.gnu.org/software/libc/manual/html_node/Getopt.html

Next three following link provide information on the `Makefile`:

http://mrbook.org/blog/tutorials/make

http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor

https://www.tutorialspoint.com/makefile/index.htm

The following link were mentioned in the text earlier:

https://www.iterm2.com

PuTTY instructions

http://smartty.sysprogs.com

https://en.wikipedia.org/wiki/Secure_Shell

http://blog.robertelder.org/what-is-ssh