CS3101: Group Project

Documentation Report

Contributors:

Oindrila Sarkar (21MS180) & Sankarshan Mondal (21MS229)

Library System

Aim

Our team aimed to create a terminal-based interactive library interface utilising concepts and data structures used in the C programming language.

System Requirements

The code was compiled using gcc.exe (MinGW.org GCC-6.3.0-1) 6.3.0. It was tested in Windows 11 23H2.

Introduction

We write the C program in a file named 'project_main.c'. The data of our library is stored in three files which are the following:

1. books.csv - stores a database of books
2. users.csv - stores a database of users and admins
3. records.csv - stores the history of actions done by users or admins

The executable file is named project_main. Alongside this, we have collated all the necessary functions in a separate header file titled 'loadfile.h'.

We have attempted to make the interface user-friendly and tolerant to user mistakes (e.g., incorrect password, incorrect book name) and re-allow input. We are aware that our program is bound to still have some shortcomings despite our rigorous efforts and have stated the scopes of improvement in a later section.

Structure of Code

The code is distributed into two files for decreasing excessively long files. The main() function is in the file named 'project_main.c'. Most of the functions are stored in a C header file named 'loadfile.h'. We would like to mention that creating this program was a wholly collaborative project and even though the functions have been modularised, both members have contributed equally - often in unseen or unrecorded ways such as help in troubleshooting or running the requisite loops. Nevertheless, we have listed the functions in both the files and the major contributor:

- project_main.c - Oindrila (body and function calls); Sankarshan (debugging and additions)
- loadfile.h

- void load_lib (char *file, book lib[]) - Oindrila
- void load_user(char *file, user db[]) - Oindrila
- void add(book l[]) - Sankarshan
- void del(book l[]) - Sankarshan
- void update(book l[]) - Sankarshan
- void query(book lib[]) - Oindrila
- void display(book library[]) - Oindrila
- int user_f(book lib[], char role_check[]) - Joint (this function includes all the user functions)
- void admin_f(book lib[]) - Sankarshan
- void disp_rec() - Sankarshan

Apart from this rough division, both parties have contributed indispensably to debugging, documentation and demonstration of the program.

Structs and Data Structures

For ease of use and coding, two data types (structs) have been defined:

```
typedef struct proto_book
{
    char name[50];
    char a_name[50];
    char publisher[20];
    char id[10];
    int cop_av;
}book;
```

- book, which has the attributes name, a_name (author), publisher, id and cop_av (number of available copies)

```
typedef struct proto_user
{
    char role[2]; //a for admin, s for student, f for faculty
    char id[50];
    char pass[50];
    int b_num;
}user;
```

- User, which has the attributes role, id, pass, b_num (number of books they have issued)

To store the databases of users and books we have created an array of these data types. Thus each element of the respective arrays has the above-mentioned attributes. The functioning of the

program and the integration of the mentioned data structures and structs shall be described in the following section.

Working of the Code

Before any user input, the program takes in two CSV files - users.csv and books.csv - and loads them into two arrays of necessary structs. These arrays are then passed to the necessary functions at necessary decision branch points to make an interactive system. In case of any error during the loading (e.g. file does not exist etc.) the program shall print a necessary error message and terminate.

After loading is successful the program prints a welcome message and asks for the username and password. If the username is not in the database, the necessary message is printed, and the user has the option to try again. If the password is incorrect, the user has three more chances to enter it correctly before the program asks them to repeat the entire login process.

Once the user has successfully logged in, the program branches internally based on the role of the user i.e., administrator, student user or faculty user. Admins are able to update, search, add or delete books to the database and an additional functionality we have added is the ability to view the records file for past usage. Users are able to search, issue and submit books. An additional functionality we have included is the ability to request a particular book. This shall be recorded in the records.csv file and can be viewed by the administrator.

Administrator Functionality

Once an admin logs in, they are prompted for their desired action: Add(A), Update(U), Delete(D), View Records(R) or Query/Search(Q). After choosing the desired action, the program branches again:

- ➔ Add allows the admin to add a new book to the database. The admin needs to enter the new book name, author, publishing company, book ID and number of copies. Once that is done, the added book’s details are displayed in the terminal.
- ➔ Delete allows the administrator to delete a book from the database by entering the book ID. After matching with the book, all the book’s details are erased and the space in the array is cleared and may be overwritten in subsequent additions.
- ➔ Update allows the administrator to update any or all (except book ID) the fields of a preexisting book. Like the add functionality, after an update is successful, the particular book's details are displayed in the terminal.
- ➔ View Records simply displays the contents of the records.csv file - which includes a history of all past actions and requests.
- ➔ Query allows the administrator to search for a particular book by entering keywords present in the book title or the author’s name. All matches are displayed.

After an unsuccessful function call (user mistake/mistyping) there is an option to login again and restart the process. After a successful login and function call, the same option is provided. When the user chooses to exit the library a thank you message is printed and the program terminates.

User Functionality

Upon login by a user, the system automatically checks whether they are a student or a faculty. The only difference in the functionality of a student and a faculty is that only faculty users are allowed to issue books with less than 3 copies. The rest of the functions (Submit(S), Query(Q) and Request(R)) are the same.

- ➔ The Issue function prompts the user to enter the name of the book they wish to issue. Once a correct title has been entered, the task is marked successful and the user is given a period of time to return the book. This time is a function of the number of copies available - specifically, twenty plus the copies available. In case a book has less than 3 copies, a student user is barred from issuing it while a faculty member is able to. An incorrect entry prompts the user to start again. The issue function also bars users from issuing books if they already have 5 books issued. This number is kept track of in the original struct user.
- ➔ Submit function also prompts the user to enter the name of the book they wish to submit. Upon successfully submitting, the database count is updated and the counter on the user decreases.
- ➔ The Request function is an added functionality which allows users to request a certain book to be added in the database. This request is recorded in the records.csv file and can be viewed and acted upon by the administrator.
- ➔ The Query function is the exact same as the one used in admin so there shall be no further elaborations.

Special Features

- Any complication in input can be rescued by simply logging in again as it refreshes the loop.
- Additional function for user - *request book*
- Additional function for admin - *view records*

Limitations

- No direct changes to the .csv files made. Can be improved once we have better practice with exact line seeking and file editing.
- Case-sensitive nature of some of the functions can be a hassle for users.
- Not very rigorous in handling all possibilities of incorrect user input. While the username, passwords and function calls are able to check for wrong inputs, some functions are unable to and simply skip to the end of the program. This can be rescued again by a new login and no infinite loops have been encountered so far.

Steps to Run the code

- ➔ Download the ‘CS3101_project’ zipped folder and extract its contents.
- ➔ Within the folder open the folder named ‘code’. This folder contains all the necessary files.
- ➔ Operator may choose to include their own CSV files for books.csv and users.csv - however the format has to be carefully maintained.
- ➔ Open the terminal in whichever OS you are using and move to the ‘code’ directory. Make sure you have **gcc(for LINUX)** or **MingGW (for Windows)** installed.
- ➔ Compile the ‘project_main.c’ file using the prompt:

```
gcc -o project_main project_main.c
```

- ➔ Run the programme using the following command:

```
./project_main
```

- ➔ The code will Run. Further instructions will be available once the code runs.

Please consult the attached video to learn more about running the code.