Computer Science 310

Project #1

The Binary Book Database

Due Date : Thursday, January 28th, 11:59 PM

50 Points

Objective

The purpose of this project is to gain experience inputting and outputting records from/to a binary file using indexed file organization. Students can gain experience using the `map` container from the STL in implementing the index.

Part I : Linux Practice (5 points)

1. Change into your CSC 310 directory, make a `Prog1` directory. Change into this directory and copy over some files for this project by typing `cp /pub/digh/CSC310/Prog1/* .`

2. Type the command `ls | wc`. Here, you're instructing the operating system to pipe the results of your file listing (`ls`) to the `wc` command. Make sure you understand the result. The use of pipes (`|`'s) will be very important for us in this course.

3. Now, let's practice our file dump command we learned. Create and save a text file with your first name in it followed by a return. Name the file `myname`. Do a file dump using the `od -xc` command. Copy and paste the contents displayed including the byte offsets into a text file named `part1.answers`.

   Look up the ASCII value of each character in your first name, and convert it to hexadecimal. Store your results in `part1.answers` and be sure you show both the ASCII value and corresponding hexadecimal value.

   Now, notice how every two characters (two bytes) in your file dump are displayed in little endian order. Little endian means the most significant value of the byte offset is shown second in the sequence.

4. Display the master file of book records, `library.dat`, using the `cat` command. Notice how we have one book record per line (i.e., each record is delimted by `'\n'`) and each field within the record is delimited by the pipe character `'|'`. Now, redisplay this file using a file dump.

5. Type `ls -l` on your master file to see its current protection mode (`pmode`). The `pmode` can be represented by a three-digit octal number based on the bits that are turned on for read/write/executable. Notice that your default is `640`. Change the (`pmode`) for this file so that it has read/write access by the owner, the group, and the world. This means we want

the bit string `110110110`, or `666` in octal. Just type `chmod 666 library.dat`. Verify your changes by re-typing `ls -l`.

6. Type the `pwd` command to determine your present working directory. Write this path down in `part1.answers`.

Part II : Processing a Master File & Building a Binary File (15 points)

Consider the following definition of a book record

```
typedef char String[25];
struct BookRec
{
        unsigned int isbn;
        String name;
        String author;
        int onhand;
        float price;
        String type;
};
```

Create a program named `create.cpp`. This program should first read from a text file and create a binary file of records of type `BookRec`. The text file will have information on zero or more books. All fields will always be in the order and type as they are defined above.

You should never have more than one `BookRec` in main memory at a time. Do not try to read the whole master file at once. Once a record has been read into memory, it should be written out to the external file `library.out`.

Along with creating the binary file, your program should handle the following potential errors in the text file.

Illegal Number – If an `isbn` is less than 1, it is considered illegal. An appropriate error message should be written to `cerr` (the unbuffered standard error device that prints output immediately). The record should then be ignored.

Number Out of Sequence – If an `isbn` is less than or equal to the previous `isbn`, an error message should be written to `cerr`, and the record should be written to `cout`. The record should be added to the output file.

Illegal OnHand or Price – If either of these fields is less than 0, it is considered illegal. An appropriate error message should be written to `cerr`, and the illegal record should be written to `cout`. The record should then be ignored.

Your program should then go through your binary file of book records in sequential order, input each record, and output it to the screen. The fields should line up in columns nicely. All records should be on a single line. You should include a leading zero where appropriate on your `isbn` numbers.

Sample Input

```
0123766891|Tom Sawyer|Twain, Mark|2|8.50|fiction
0239678325|Leaves of Grass|Whitman, Walt|8|29.99|poetry
0243578325|Romeo and Juliet|Shakespeare, William|6|4.99|drama
0249638325|Great Gatsby|Fitzgerald, F. Scott|0|5.99|fiction
0259648323|Scarlet Letter|Hawthorne, Nathaniel|8|4.78|fiction
1229648991|Whisper of the River|Sams, Ferrol|4|21.95|fiction
1239678325|Moby Dick|Melville, Herman|2|13.98|fiction
2119674425|Last of the Mohicans|Cooper, James Fenimore|1|8.75|fiction
2269572525|Odyssey|Homer|5|9.99|fiction
3389678325|Christmas Carol|Dickens, Charles|-6|12.50|fiction
3393578325|Les Miserables|Hugo, Victor|2|19.98|fiction
3391679915|Heart of Darkness|Conrad, Joseph|0|14.45|fiction
-369957832|Animal Farm|Orwell, George|1|10.00|fiction
3791527325|Canterbury Tales|Chaucer, Geoffrey|1|20.00|drama
3995783225|Old Man and the Sea|Hemingway, Ernest|3|9.95|fiction
```

Sample Output

```
> Negative amount onhand on line 10 of data file - record ignored.
 3389678325      Christmas Carol        Dickens, Charles  -6  12.50   fiction
> Isbn number out of sequence on line 12 of data file.
 3391679915    Heart of Darkness          Conrad, Joseph   0  14.45   fiction
> Illegal isbn number encountered on line 13 of data file - record ignored.

 0123766891           Tom Sawyer             Twain, Mark   2   8.50   fiction
 0239678325      Leaves of Grass           Whitman, Walt   8  29.99    poetry
 0243578325     Romeo and Juliet    Shakespeare, William   6   4.99     drama
 0249638325         Great Gatsby    Fitzgerald, F. Scott   0   5.99   fiction
 0259648323       Scarlet Letter    Hawthorne, Nathaniel   8   4.78   fiction
 1229648991 Whisper of the River            Sams, Ferrol   4  21.95   fiction
 1239678325            Moby Dick        Melville, Herman   2  13.98   fiction
 2119674425 Last of the Mohicans  Cooper, James Fenimore   1   8.75   fiction
 2269572525              Odyssey                   Homer   5   9.99   fiction
 3393578325       Les Miserables            Hugo, Victor   2  19.98   fiction
 3391679915    Heart of Darkness          Conrad, Joseph   0  14.45   fiction
 3791527325     Canterbury Tales       Chaucer, Geoffrey   1  20.00     drama
 3995783225  Old Man and the Sea       Hemingway, Ernest   3   9.95   fiction
```

Part III : The Database (30 points)

Write a program to handle the inventory for a small book company using keyed sequential records. The company has determined that the `BookRec` definition from earlier will adequately represent the books they have in stock.

The current inventory of books stocked will be kept in a master binary file (e.g., the one you created one in Part II – `library.out`). You may assume that ISBN is a unique key (so there will

not be more than one record with the same ISBN).

You will have a binary transaction file that is formatted similarly to the master file with one additional field :

```
enum TransactionType {Add, Delete, ChangeOnhand, ChangePrice};

struct TransactionRec
{
     TransactionType ToDo;
     BookRec B;
};
```

Your program should perform the transaction as specified by the additional field `ToDo`. You may assume the transaction file is also sorted in ascending order by ISBN.

Your program should go through the master file and transaction file and create two new files, a new master file and an error file named `ERRORS`. The new master file is another binary file of book records and the error file is a text file. Once your new master file has been created, your program should read in each of the records contained within it, and display them to the screen.

Adding Records (when the `ToDo` field is `Add`)

If the ISBN does not already exist, add a new record to the new master, containing all of the other information in the `TransactionRec`, in the appropriate position. If it does exist, print an error message in the following format :

Error in transaction number xxx: cannot add—duplicate key yyyyyyyyyy

where xxx is the number of the transaction record causing the error (where the first record in the transaction file is 1, the second is 2, etc.) and yyyyyyyyyy is the ISBN.

An add transaction may occur before other transactions for the same ISBN. This is legal and the changes/deletes should occur on the new record, before it goes into the new master file.

Deleting Records (when the `ToDo` field is `Delete`)

If the ISBN exists, the record should not be added to the new master file. If the ISBN does not exist, print an error message in the following format :

Error in transaction number xxx: cannot delete—no such key yyyyyyyyy

where xxx is the number of the transaction record causing the error and yyyyyyyyyy is the ISBN.

Changing the On Hand Amount (when the `ToDo` field is `ChangeOnhand`)

If the ISBN exists, change the `Onhand` field by adding the `Onhand` field of the transaction record to it. If this creates a negative count, print an error message in the following format :

Error in transaction number xxx: count = zz for key yyyyyyyyyy

where xxx is the number of the transaction record causing the error, zzz would be the final (negative) count field, and yyyyyyyyyy is the ISBN.

Change the negative count to 0 after printing the error message and before storing the record.

If the ISBN does not exist, print an error message in the following format :

Error in transaction number xxx: cannot change count—no such key yyyyyyyy

where xxx is the number of the transaction record causing the error and yyyyyyyyyy is the ISBN.

Changing the Price (when the `ToDo` field is `ChangePrice`)

If the ISBN exists, change the price field to the price field of the transaction record.

If the ISBN does not exist, print an error message in the following format :

Error in transaction number xxx: cannot change price—no such key yyyyyyyyy

where xxx is the number of the transaction record causing the error and yyyyyyyyyy is the ISBN.

Database Details

Your program should be set up to take the name of the master file, transaction file, and new master file from the command line (in this order). Your program should be named `update.cpp`.

The original master file should not be modified in any way during your program. Make a copy of it from within your program that you can use for appending new records. (e.g., `system("cp library.out copy.out");` Delete your copy from within your program.

You should not have more than two book records or one transaction record in main memory at a time. Do not try to read in the whole master file or multiple transactions at once.

You may use a `map` container from the STL to build your index of isbn fields and corresponding byte offsets.

You will not be able to read the master or transaction files since they are in binary. However, if you view them in `vi`, the titles should stand out. I have provided a sample master file `library.out`, a sample transaction file `transact.out`, and a new master file `update.out`. Here is a summary

of the data contained in them translated to human readable form.

Sample Master File

```
0123766891           Tom Sawyer             Twain, Mark   2   8.50   fiction
0239678325      Leaves of Grass           Whitman, Walt   8  29.99    poetry
0243578325     Romeo and Juliet    Shakespeare, William   6   4.99     drama
0249638325         Great Gatsby    Fitzgerald, F. Scott   0   5.99   fiction
0259648323       Scarlet Letter    Hawthorne, Nathaniel   8   4.78   fiction
1229648991 Whisper of the River            Sams, Ferrol   4  21.95   fiction
2119674425 Last of the Mohicans  Cooper, James Fenimore   1   8.75   fiction
3393578325       Les Miserables            Hugo, Victor   2  19.98   fiction
3791527325     Canterbury Tales       Chaucer, Geoffrey   1  20.00     drama
```

Sample Transaction File

```
Add 0123766891              Tom Sawyer            Twain, Mark   2   8.50   fiction
Delete 0243578325     Romeo and Juliet   Shakespeare, William   6   4.99   drama
ChangeOnhand 0259648323 Scarlet Letter   Hawthorne, Nathaniel   4   4.78   fiction
Add 1239678325               Moby Dick       Melville, Herman   2  13.98   fiction
Add 2269572525                 Odyssey                 Homer   5   9.99   fiction
Add 3391679915       Heart of Darkness         Conrad, Joseph   0  14.45   fiction
Add 3995783225     Old Man and the Sea      Hemingway, Ernest   3   9.95   fiction
```

Sample Output (of updated master file sent to screen)

```
0123766891           Tom Sawyer             Twain, Mark   2   8.50   fiction
0239678325      Leaves of Grass           Whitman, Walt   8  29.99    poetry
0249638325         Great Gatsby    Fitzgerald, F. Scott   0   5.99   fiction
0259648323       Scarlet Letter    Hawthorne, Nathaniel  12   4.78   fiction
1229648991 Whisper of the River            Sams, Ferrol   4  21.95   fiction
1239678325            Moby Dick       Melville, Herman   2  13.98   fiction
2119674425 Last of the Mohicans  Cooper, James Fenimore   1   8.75   fiction
2269572525              Odyssey                 Homer   5   9.99   fiction
3391679915    Heart of Darkness         Conrad, Joseph   0  14.45   fiction
3393578325       Les Miserables            Hugo, Victor   2  19.98   fiction
3791527325     Canterbury Tales       Chaucer, Geoffrey   1  20.00     drama
3995783225  Old Man and the Sea      Hemingway, Ernest   3   9.95   fiction
```

Sample Output (of error file)

```
Error in transaction number   1: cannot add---duplicate key 0123766891
```

You should write a short program to generate additional transaction files that can be used as test cases. Obvious choices are special cases: data which causes each of the kinds of errors, each of the operations, empty master and transactions files, multiple transactions for a single master file record, etc. Although you may not share code with classmates, feel free to exchange test data.

Analysis & Design Discussion for Part III

In one page, address each of the following points :

1. The goal or problem solved by your client program.
2. A clear specification of all input and output used by client program. All input/output files should be described.
3. A brief outline of the algorithm used by your client program. The reader should be able to code your client file after reading your algorithm. Remember that algorithms avoid code references. Mention any comments on the efficiency of your algorithm and/or any steps you have taken to make it more efficient.

Store this in a file named `README` in your current directory.

Finishing Up

Create a `typescript` file in your project directory using the following steps:

1. display your client program by typing `cat update.cpp`
2. provide a sample compilation by typing `c++ update.cpp`
3. provide a sample run with input
4. exit typescript by typing `exit`

Submit

Copy over your files for grading on /scratch. Remember to replace `last_fm` with your cobra login name below.

```
cp *.cpp /scratch/csc310/last_fm/Prog1
cp *.h /scratch/csc310/last_fm/Prog1
cp typescript /scratch/csc310/last_fm/Prog1
cp README /scratch/csc310/last_fm/Prog1
```