

Project1 Report

Haiyang Yun

ID:1233970

1. The basic decision on this project:

Page Organization:

To create a buffer manager, I should decide the page organization at first. The first part of file is a metadata space. The size of this metadata is same as frame size, which is 4096. The space following this metadata is for the true data. And I arrange the first byte of every block as a flag to indicate this block is disposed or not. So, every block “size” is FRAMESIZE+sizeof(char)=4097 actually. But the true size of every block is still 4096 bytes.

File metadata:

As I mentioned at the first part. The metadata is stored in the first 4096 bytes space. This metadata is for storing the last-used block of this file. Since we have functions aim at getting the next block and “this” block of a certain file. It is necessary to have this kind of metadata.

Buffer Pool Structure:

In the BM_init function, I generate a 100 elements

structure array, which name is bufferpool. This array is my buffer pool.

Buffer Pool Replacement Policy:

I use MRU(most recently used) as my replacement policy. Since it is easier to be implemented in C code. I just need a extern variable to store the number of last-used.

2. Challenges in developing.

In developing my project code, my challenges mostly caused by unfamiliar with C language and the file organization.

Since this is the first time I use C language to develop a project. I do not know how to use the file pointer to read data from file and put them into my buffer pool.

Fortunately, I found some examples in StackOverflow and GitHub. I tried my best to use all these file stream functions though I don't understand them exactly. But the result should be good.

Another challenge is the file organization. Since I do not have an overview of my buffer manager, I have to

change my file's organization when facing different kinds of problem. For example, I didn't define any extern variable at the beginning. It is hard to get the last-used block when implementing MRU policy. So I have to re-define my extern variable.

Next and the last challenge is about the complier. Because I use XCODE in my whole develop. And Xcode has a different compiler with gcc. So, at some circumstance, I can build and compile my testmain in Xcode, but fail compiling in terminal by using gcc(the makefile). And I cannot solve this problem till now.

3. The test result.

I use the same testmain as professor provide with. After I finished all my functions, I test my function part by part. At first I give a return 0 at line 38.

```

7
8
9 //populates an existing block with some random data
10 void populate_block(block* blockPtr);
11
12
13 int main(int argc, char const *argv[])
14 {
15     int errc = 0;
16     /*
17     bm init
18     */
19     BM_init();
20
21     /*
22     create a file
23     insert pages filled with data
24     close a file
25     */
26     BM_create_file("testing.dat");
27     fdf = BM_open_file(fdopen(fd, "w+"), "testing.dat");
28     BM_alloc_block(fd);
29     block* blockPtr=NULL;
30     BM_get_first_block(fd, &blockPtr); //this function will assign blockPtr to a valid block from the BP
31     populate_block(blockPtr);
32     BM_unpin_block(blockPtr); //just made changes
33     //make sure the loaded frame can actually be swapped out (when the corresponding file is closed).
34     //we do not expect to close a file with pinned blocks
35     BM_unpin_block(blockPtr);
36     assert(blockPtr->pinCount==0); //pin count should be 0
37     BM_close_file(fd);
38     return 0;
39
40     /*
41     bm open file
42     loop this stuff
43     bm alloc block

```

Then I build and run the project. The result is :

```

Buildtime (6) Runtime
DatabaseProject1 6 issues
Dependency Analysis Warning
Warning: no rule to process file '/Users/YHY/Desktop/DatabaseProj/DatabaseProject1/Database...
Semantic Issue
Comparison between pointer and integer ('char*' and 'int') projectA.c
Comparison between pointer and integer ('char*' and 'int') projectA.c
Code will never be executed testmain.c
Comparison between pointer and integer ('char*' and 'int') projectA.c
Unused Entity Issue
Unused variable 'errc' testmain.c

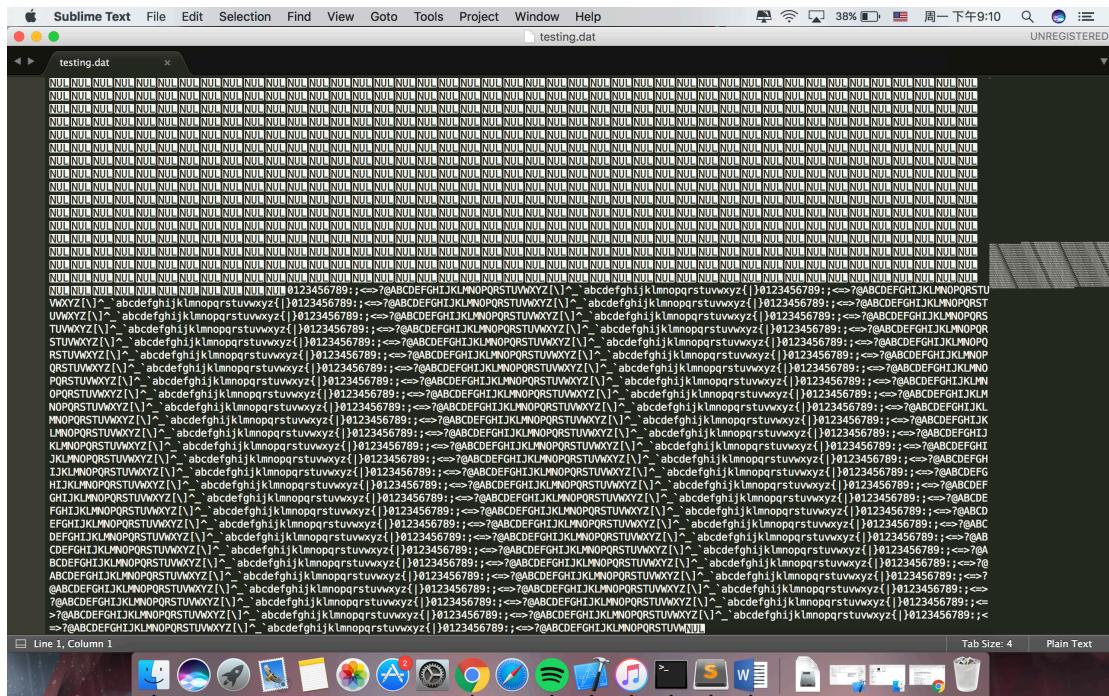
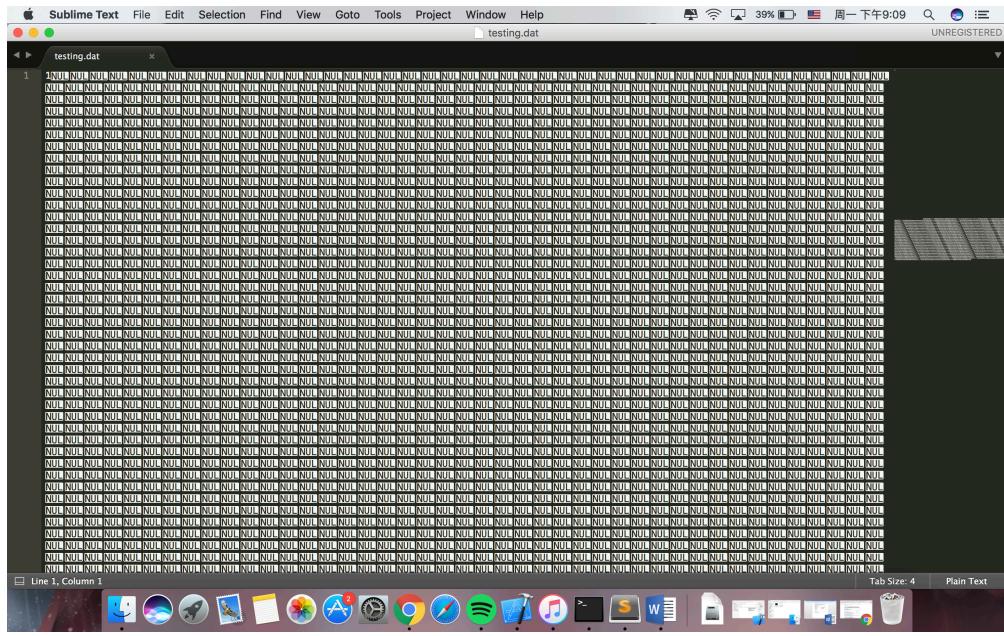
```

```

Calling BM_create_file with name: testing.dat
Calling BM_open_file with name: testing.dat
Attempting to allocate block in file 3
Attempting to read first block from file 3
The block id of first block is 1
metadata is: 1
Unpinning block
Attempting to close file: 3
File has been closed successfully
Program ended with exit code: 0

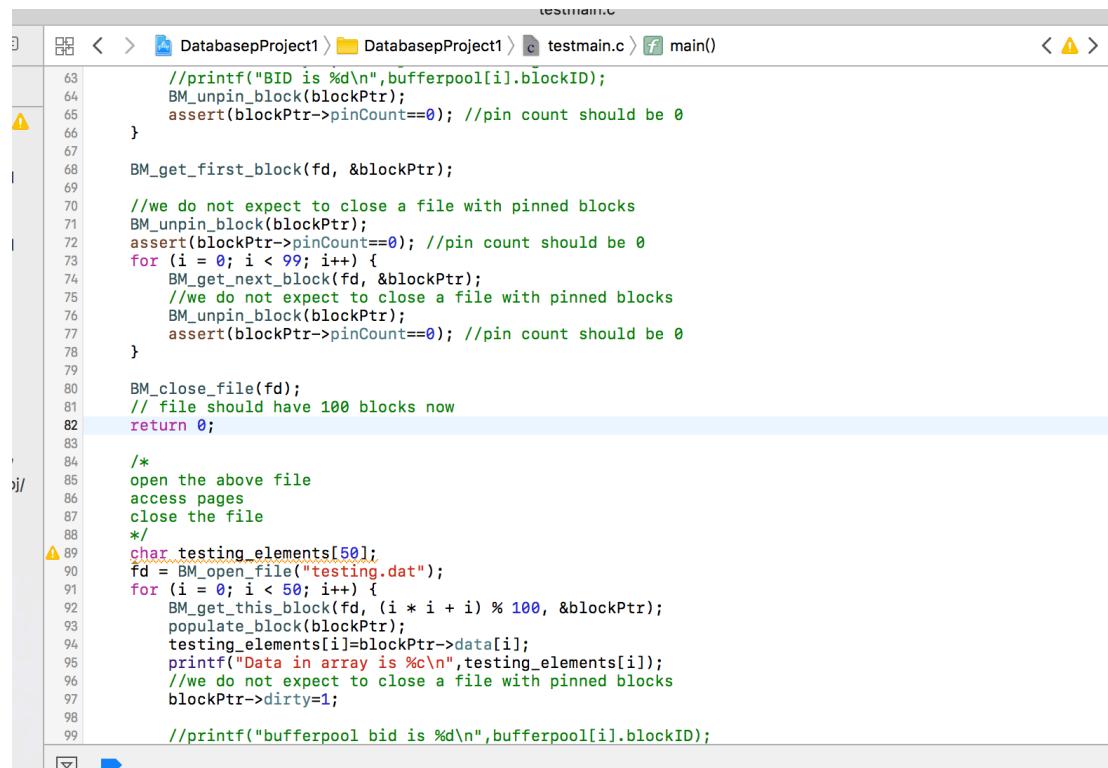
```

The data in the file is :



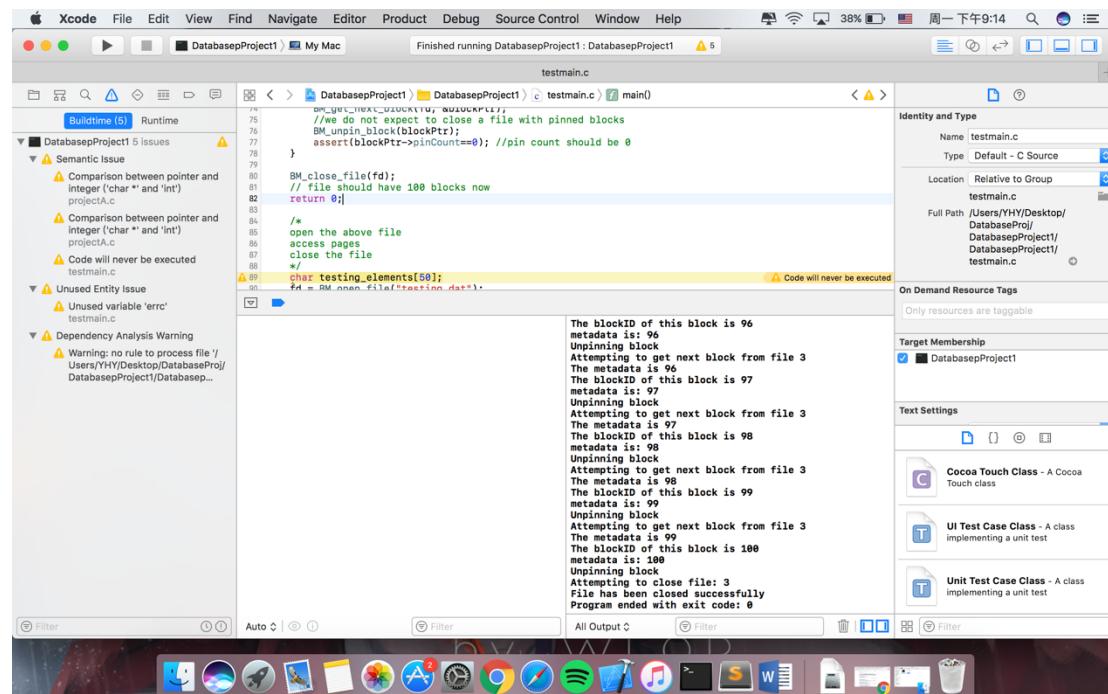
It shows that the block has been put into my file. The file part is metadata with a counting number shows the last-used block. The following part is the data generated by populate_block function.

Then I give return 0 at line 82.



```
testmain.c
DatabaseProject1 DatabaseProject1 c main()
63     //printf("BID is %d\n",bufferpool[i].blockID);
64     BM_unpin_block(blockPtr);
65     assert(blockPtr->pinCount==0); //pin count should be 0
66 }
67
68 BM_get_first_block(fd, &blockPtr);
69
70 //we do not expect to close a file with pinned blocks
71 BM_unpin_block(blockPtr);
72 assert(blockPtr->pinCount==0); //pin count should be 0
73 for (i = 0; i < 99; i++) {
74     BM_get_next_block(fd, &blockPtr);
75     //we do not expect to close a file with pinned blocks
76     BM_unpin_block(blockPtr);
77     assert(blockPtr->pinCount==0); //pin count should be 0
78 }
79
80 BM_close_file(fd);
81 // file should have 100 blocks now
82 return 0;
83
84 /*
85 open the above file
86 access pages
87 close the file
88 */
89 char testing_elements[50];
90 fd = BM_open_file("testing.dat");
91 for (i = 0; i < 50; i++) {
92     BM_get_this_block(fd, (i * i + i) % 100, &blockPtr);
93     populate_block(blockPtr);
94     testing_elements[i]=blockPtr->data[i];
95     printf("Data in array is %c\n",testing_elements[i]);
96     //we do not expect to close a file with pinned blocks
97     blockPtr->dirty=1;
98
99 //printf("bufferpool bid is %d\n",bufferpool[i].blockID);
```

The result is



Buildtime (5) Runtime

DatabaseProject1 5 issues

- Semantic Issue
 - Comparison between pointer and integer ('char *' and 'int')
 - Comparison between pointer and integer ('char *' and 'int')
 - Code will never be executed
- Unused Entity Issue
 - Unused variable 'errc'
- Dependency Analysis Warning
 - Warning: no rule to process file '/Users/YHY/Desktop/DatabaseProj/DatabaseProject1/Database...'

testmain.c

```
The blockID of this block is 96
metadata is: 96
Unpinning block
Attempting to get next block from file 3
The metadata is: 96
The blockID of this block is 97
metadata is: 97
Unpinning block
Attempting to get next block from file 3
The metadata is: 97
The blockID of this block is 98
metadata is: 98
Unpinning block
Attempting to get next block from file 3
The metadata is: 98
The blockID of this block is 99
metadata is: 99
Unpinning block
Attempting to get next block from file 3
The metadata is: 99
The blockID of this block is 100
metadata is: 100
Unpinning block
Attempting to close file: 3
File has been closed successfully
Program ended with exit code: 0
```

And the data in file:

The first part is metadata. And all the following space is filled with data.

Then I give a return at line 122:

The screenshot shows the Xcode IDE interface with the following details:

- Project Navigator:** Shows 'DatabaseProject1' with 5 issues: 1 Semantic Issue, 1 Unused Entity Issue, 1 Dependency Analysis Warning, and 3 Code will never be executed.
- Code Editor:** The file `testmain.c` is open, containing C code related to a buffer pool and file operations. Several lines of code are annotated with yellow triangles and text boxes:
 - Line 96: `blockPtr->dirty=1;`
 - Line 98: `//print("bufferpool bid is %d\n",bufferpool[i].blockID);`
 - Line 100: `BM_unpin_block(blockPtr);`
 - Line 101: `assert(blockPtr->pinCount==0); //pin count should be 0`
 - Line 103: `BM_close_file(fd);`
 - Line 109: `/*`
 - Line 110: `bm open specific file (checks that its opening an existing file not making a new one)`
 - Line 111: `bm get blocks and examine data`
 - Line 112: `bm close file`
 - Line 114: `fd = BM_open_file("testing.dat");`
 - Line 115: `for (i = 0; i < 50; i++) {`
 - Line 116: `BM_get_this_block(fd, (i * i + i) % 100, &blockPtr);`
 - Line 117: `assert(testing_elements[i]==blockPtr->data[i]); //assert we read the same bytes we wrote`
 - Line 118: `BM_unpin_block(blockPtr);`
 - Line 119: `assert(blockPtr->pinCount==0); //pin count should be 0`
 - Line 120: `}`
 - Line 121: `BM_close_file(fd);`
 - Line 122: `return 0;`
 - Line 124: `/*`
 - Line 125: `open the above file`
 - Line 126: `remove pages`
 - Line 127: `close the file`
 - Line 128: `*/`
 - Line 129: `fd = BM_open_file("testing.dat");`
 - Line 130: `for (i = 0; i < 100; i++) {`
 - Line 131: `BM_dispose_block(fd, i);`
 - Line 132: `}`
- Annotations:** A yellow triangle points to the line `assert(blockPtr->pinCount==0);` with the text "Code will never be executed". Another yellow triangle points to the line `BM_close_file(fd);` with the text "Unused Entity Issue". A third yellow triangle points to the line `assert(blockPtr->pinCount==0);` with the text "Dependency Analysis Warning".
- Callouts:** A callout box at the bottom right shows the metadata for a block: "The metadata is 95", "The blockID of this block is 96", "metadata is: 96", "Unpinning block", "Attempting to get next block from file 3", and "The metadata is 96".
- Right Sidebar:** Shows sections for Identity and Type, Location, Full Path, On Demand Resource, Target Membership, Text Settings, and Unit Test Case.

The result is:

```

/*
bm open specific file (checks that its opening an existing file not making a new one)
bm get blocks and examine data
bm close file
*/

```

```

fd = BM_open_file("testing.dat");
for (i = 0; i < 50; i++) {
    BM_get_this_block(fd, (i * i + i) % 100, &blockPtr);
    assert(testing_elements[i]==blockPtr->data[i]); //assert we read the same bytes we wrote
    BM_unpin_block(blockPtr);
    assert(blockPtr->pinCount==0); //pin count should be 0
}

```



```

metadata is: 63
Unpinning block
Attempting to get block 56 from file 3
The blockID of this block is 57
metadata is: 57
Unpinning block
Attempting to get block 52 from file 3
The blockID of this block is 53
metadata is: 53
Unpinning block
Attempting to get block 50 from file 3
The blockID of this block is 51
metadata is: 51
Unpinning block
Attempting to close file: 3
File has been closed successfully
Program ended with exit code: 0

```

It means the data we wrote is same as we read. So I pass this part.

At last I run the whole test code. And the result is :

```

50     /*+
51      all throughout print some errors if there are any.
52      */
53      return 0;
54 }

55

56 void populate_block(block* blockPtr) {
57
58     //assumes a block retrieved from BP, updates its contents
59     assert(blockPtr!=NULL);
60
61     char* raw_data = malloc(sizeof(char) * FRAMESIZE));
62     int i;
63     char tmp ;
64     for (i = 0; i < FRAMESIZE; i++){
65         tmp=i%78+'0';
66         raw_data[i] = tmp;
67     }
68     //here we copy to the contents of the block
69     memcpy(blockPtr->data, raw_data, FRAMESIZE);
70     free(raw_data);
71 }

```



```

Attempting to dispose of block 91 from file 3
Attempting to dispose of block 92 from file 3
Attempting to dispose of block 93 from file 3
Attempting to dispose of block 94 from file 3
Attempting to dispose of block 95 from file 3
Attempting to dispose of block 96 from file 3
Attempting to dispose of block 97 from file 3
Attempting to dispose of block 98 from file 3
Attempting to dispose of block 99 from file 3
Attempting to close file: 3
File has been closed successfully
Calling BM_open_file with name: testing.dat
Attempting to read first block from file 3
The block id of first block is 1
metadata is: 1
Unpinning block
Attempting to dispose of block 0 from file 3
Attempting to close file: 3
File has been closed successfully
Program ended with exit code: 0

```

The result of the whole project is correct.

4. More discussion:

The most challenging and interesting part is use pointer to assign data between buffer pool and file. I spend most of my time on doing and debugging this part. And I still make the result perfect.