1. Environment

   OS: windows 10

   VS version: VS 2017

   CUDA version: 9.2

   GPU: GTX 1050 Ti

2. Execution steps of the program

   a) Use VS 2017 to open the project
   b) Use ctrl+f7 to compile the .cu files in the folder
   c) Press ctrl+f5 to run the program
   d) Wait for the result

3. Design of the program

   a) FCB entry

      Each FCB entry will record the information of the corresponding files and each of them is of size 32 bytes.

      i. The 0-19 bytes will be used to record the file name (including the '\0' at the end).
      ii. The 20 and 21 bytes will be used to record the block address of the file. Notice that, the total block number is 1024KB/32B = 32K, which is less than the maximum number represented with 2 bytes, that is, 2^16.
      iii. The 22 and 23 bytes will be used to store the size of the file, this will be sufficient since the maximum size of the file is 1024. Notice that, the 22th byte is also used as the valid bit, that is, if the current FCB is invalid, it will be set to 0xff, otherwise it will be valid.
      iv. The 24 and 25 bytes will be used to store the created time of the file, the program will record the relative time of the files, that is the created order of the files. Therefore, the created time is at most 1023, which is less than the maximum number represented by 2 bytes.
      v. The 26 and 27 bytes will be used to store the modified time of the file,

like the created time, the program will record the relative time of the files, that is the modified order of the files. Therefore, the modified time is at most 1023, which is less than the maximum number represented by 2 bytes.

b) Global variables:

i. u32 gtime:

This is used to record the relative created time for the next file, which means it is just equal to the current number of exist files.

ii. u32 global_storage_end:

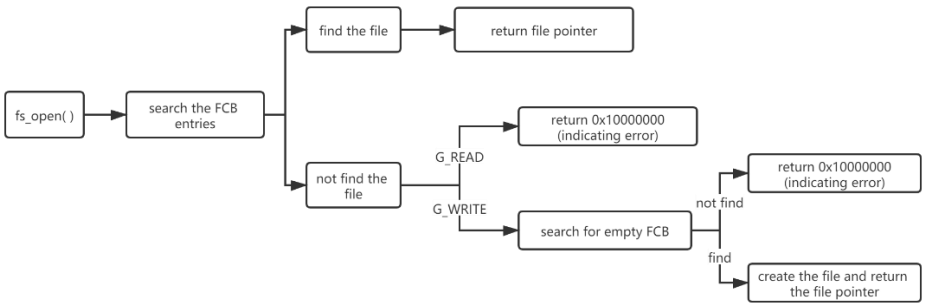
This is used to record the address of the next memory block after the last used block in the file system, since the memory is allocated continuously in the file content, this is just the first free memory block in the file system.

c) u32 fs_open(FileSystem *fs, char *s, int op):

i. function:

It will return the file pointer of the file with the given name. If the file is not found and in G_WRITE mod, it will create a new file.

ii. diagram of the basic logic:



iii. implementation:

The implementation is mainly following the logic of the diagram shown above. Notice that, when create the files, it will do the basic initialization

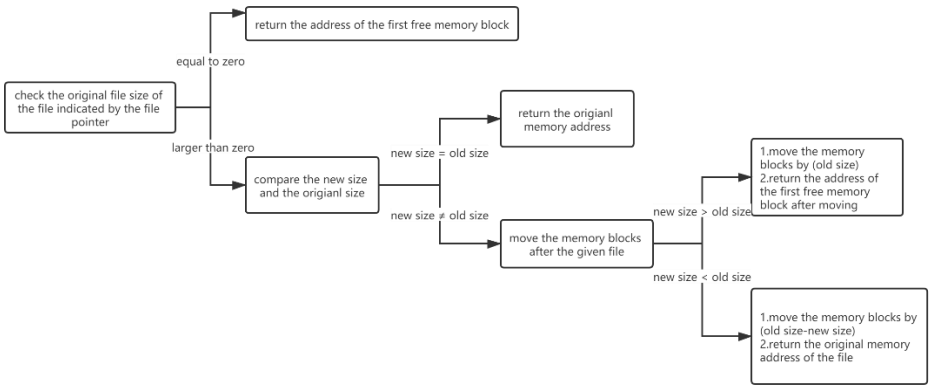
of the FCB entry of the files.

d) u32 fs_mount(FileSystem *fs, int new_size, u32 fp):

i. function:

It will move all the memory blocks after the file indicated by the file pointer properly according to the new size of the file. After that, it will return the new memory address for that file pointer.

ii. diagram of the basic logic:



iii. implementation:

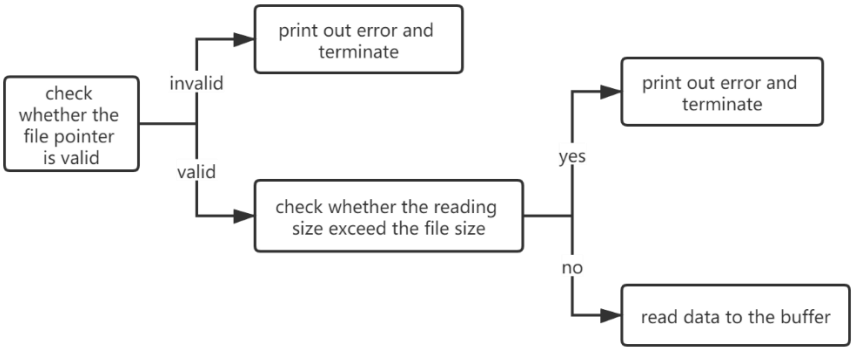
The implementation mainly following the diagram shown above. Notice that for memory blocks, the program will move them together instead of moving one by one and the program won't clear the remaining memory which are previously belong to some files since the global memory pointer will tread them as free memory and will directly rewrite them when using. For superblock, the program will move them one by one during the modification of the FCB of files whose memory block lies behind that of the file indicated by the file pointer.

e) void fs_read(FileSystem *fs, uchar *output, u32 size, u32 fp):

i. function:

This function will read the data from the file to the given output buffer.

ii. diagram of the basic logic:



iii. implementation:

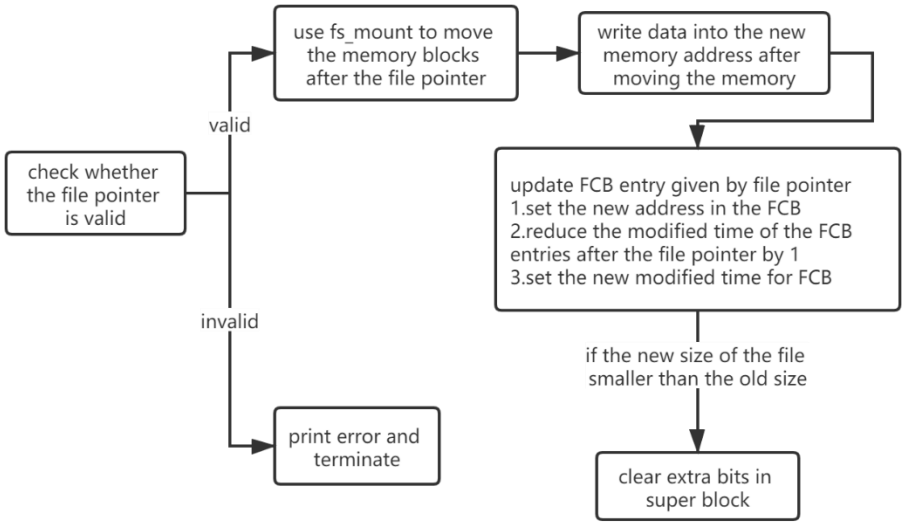
The implementation basically follows the diagram above.

f) u32 fs_write(FileSystem *fs, uchar* input, u32 size, u32 fp):

i. function:

It will write the data to the file given by the file pointer.

ii. diagram of the basic logic:



iii. implementation:

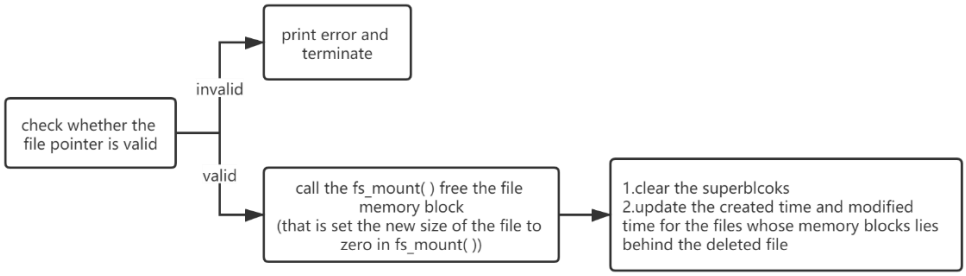
The implementation basically follows the diagram shown above.

g) void fs_gsys(FileSystem *fs, int op, char *s):

i. function:

If the op equals RM (defined as 2), it will delete the file given by the file name and do the memory compaction if necessary.

ii. diagram of the basic logic:



iii. implementation:

The implementation basically follows the diagram above. Notice that when deleting a file, the memory blocks of the file will be deleted and to maintain the continuous allocation, the program will just move all the memory blocks after the file forward, and which is same as the function of fs_mount( ) if we set the new size of the file to zero. Therefore, the program will directly use the fs_mount to clear the memory blocks of the file. Also, since the created and modified times for the files are relative time in the program, the program needs to modifify the created and modified time of the FCB entries to make sure they lie in the 0 to 1023.

h) void fs_gsys(FileSystem *fs, int op):

i. function:

If the op equals LS_D, the program would list all the files by the decreasing order of their modified time. If the op equals LS_S, the program would list all the files by the decreasing order of their file size. If two files have the same size, it will first list the file created first.

ii. Implementation:

1) LS_D:

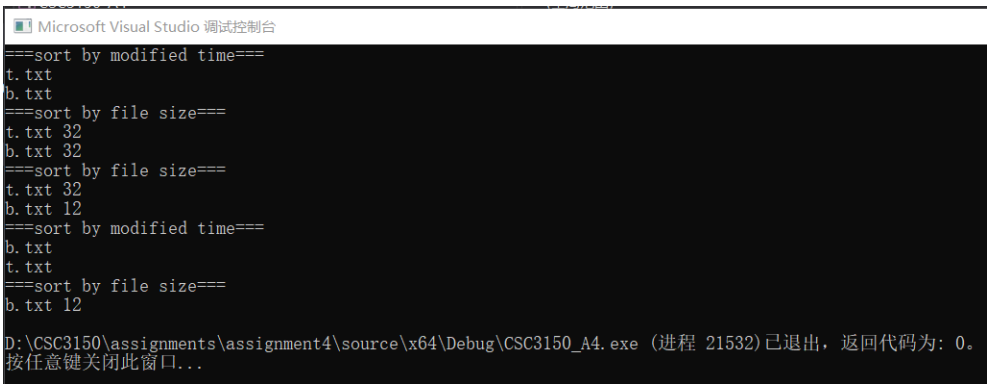
Since the modified and created time for the files in the program is the relative time, which is just the order of the time among files. Therefore, to list the files order by the modified time, we can just use for-loop to go through the FCB entries by gtime (shown in (b) Global variables) times from gtime-1 to 0 (since the created and modified time for the files are just the orders, the gtime for the next creating file is just the total number of the current files), and each time just print out the file whose created time equals the current order the iteration.

2) LS_S:

Since the memory limit of the program, the program will not use extra memory to do the sorting. Instead, the program would also go through the FCB entries gtime times, and in each iteration, for example in the ith iteration, it will just find the ith largest element and print it out instead of storing it in an array. Notice that, it the files have same size, the program would treat the one with smaller created time to be larger.

4. Outputs:

a) Test case 1



b) Test case 2

```
Microsoft Visual Studio 调试控制台
===sort by modified time===
t.txt
b.txt
===sort by file size===
t.txt 32
b.txt 32
===sort by file size===
t.txt 32
b.txt 12
===sort by modified time===
b.txt
t.txt
===sort by file size===
b.txt 12
===sort by file size===
*ABCDEFGHIJKLMNOPQR 33
)ABCDEFGHIJKLMNOPQR 32
(ABCDEFGHIJKLMNOPQR 31
'ABCDEFGHIJKLMNOPQR 30
&ABCDEFGHIJKLMNOPQR 29
%ABCDEFGHIJKLMNOPQR 28
$ABCDEFGHIJKLMNOPQR 27
#ABCDEFGHIJKLMNOPQR 26
"ABCDEFGHIJKLMNOPQR 25
!ABCDEFGHIJKLMNOPQR 24
b.txt 12
===sort by modified time===
*ABCDEFGHIJKLMNOPQR
)ABCDEFGHIJKLMNOPQR
(ABCDEFGHIJKLMNOPQR
'ABCDEFGHIJKLMNOPQR
&ABCDEFGHIJKLMNOPQR
b.txt

D:\CSC3150\assignments\assignment4\source\x64\Debug\CSC3150_A4.exe (进程 6844)已退出，返回代码为: 0。
按任意键关闭此窗口...
```

c) Test case 3 (partial)

```
Microsoft Visual Studio 调试控制台
BA 40
AA 39
@A 38
?A 37
>A 36
=A 35
<A 34
*ABCDEFGHIJKLMNOPQR 33
;A 33
)ABCDEFGHIJKLMNOPQR 32
:A 32
(ABCDEFGHIJKLMNOPQR 31
9A 31
'ABCDEFGHIJKLMNOPQR 30
8A 30
&ABCDEFGHIJKLMNOPQR 29
7A 29
6A 28
5A 27
4A 26
3A 25
2A 24
b.txt 12

D:\CSC3150\assignments\assignment4\source\x64\Debug\CSC3150_A4.exe (进程 19012)已退出，返回代码为: 0。
按任意键关闭此窗口...
```

d) snapshot.bin

i. test 1:

snapshot.bin | data.bin | file_system.cu | user_program.cu | main.cu

```
00000000  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000010  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

ii. test 2:

snapshot.bin | data.bin | file_system.cu | user_program.cu | main.cu

```
00000000  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000010  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

iii. test 3 (partial):

snapshot.bin data.bin file_system.cu user_program.cu main.cu

```
00000000  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000010  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

```
000003e0  00 00 00 00 00 00 00 00  FA E5 17 F8 A2 DC 72 AF  ..............r.
000003f0  4B A0 28 C0 C3 18 25 EE  E6 9F F4 44 5A 7E 8E E9  K.(...%....DZ~..
00000400  95 C5 E4 24 E3 74 29 DE  D9 BF 57 FC 9D CA AC E8  ...$.t)...W.....
00000410  6B 54 2A AE EB CE 1D D3  EE 12 97 49 90 A5 B2 A6  kT*........I....
00000420  6B 97 CA 50 8C 73 AE 66  34 07 63 D1 D1 10 3A BC  k..P.s.f4.c...:.
00000430  64 E3 6B 51 B3 88 A4 22  1A BB 6B AA 61 9D 51 CC  d.kQ..."..k.a.Q.
00000440  B5 9C 9C 42 10 4C A8 44  53 0D 95 A4 9C 50 E0 81  ...B.L.DS....P..
00000450  B3 CB D2 67 54 F6 89 ED  B2 74 98 14 92 6A 60 48  ...gT....t...j`H
00000460  07 FD 8A 96 4A 33 DB 1D  BF F0 41 5D C0 22 DE 75  ....J3....A]."u
00000470  ED 31 5C C1 28 66 AF 5A  DA C7 ED EC B1 4E 35 38  .1\.(f.Z.....N58
00000480  CB 3F CF 95 F2 2B 32 B2  1C 73 10 DD 95 6E 53 03  .?...+2..s...nS.
00000490  1F AF 44 C6 16 F3 21 71  3C 8E DD ED 5D 14 27 29  ..D...!q<...].')
000004a0  53 F6 3F C5 22 71 79 BD  E5 09 1B FA F7 ED 7E 17  S.?."qy.......~.
000004b0  1E C2 DE B3 B7 00 A4 F3  8F 83 61 EC 17 88 95 E9  ..........a.....
000004c0  FE D4 B0 21 47 2A 5F AC  33 7A A7 AA E8 26 C2 07  ...!G*_.3z...&..
000004d0  E9 A1 BA 21 21 DF 94 30  63 F5 1D 7A FE 33 64 FD  ...!!..0c..z.3d.
000004e0  87 15 9F CE BE FE FA 72  F8 A3 1D 61 49 DF 68 33  .......r...aI.h3
000004f0  81 A3 D3 23 83 E7 53 E6  5E F0 E0 5D 24 C4 5B AB  ...#..S.^..]$.[.
00000500  DA 7A FA 19 F8 F5 8B 72  19 A9 D3 63 09 3D 16 0B  .z.....r...c.=..
00000510  60 EA 2E E3 52 81 4A B0  72 2B 0E 16 EF E9 42 4A  `...R.J.r+....BJ
00000520  64 BC 64 DD 32 6F 50 4C  98 24 AF A2 61 45 2D 41  d.d.2oPL.$..aE-A
00000530  AF 5B 25 03 5C EE B3 4F  99 42 65 0A 2C 27 54 10  .[%.\..O.Be.,'T.
00000540  E3 38 ED 17 28 3E 63 C0  E2 92 63 44 D7 90 06 88  .8..(>c...cD....
00000550  6B 2B 0B C8 9A BE 18 34  80 FC 3E 2C 25 13 3D 09  k+.....4..>,%.=.
00000560  CA AA 20 F2 69 03 B4 4C  95 97 10 ED A8 16 F5 14  .. .i..L........
00000570  42 01 5C DC 3F F3 90 40  F1 CF 6C 17 E2 A9 9F AD  B.\.?..@..l.....
00000580  55 C0 A1 BE 43 D5 8A D9  6D 9A 47 95 B1 3D 2A F3  U...C...m.G..=*.
00000590  BD 86 50 7C 7B E0 BC 6D  30 2A 04 92 53 A3 C0 28  ..P|{..m0*..S..(
000005a0  E3 E1 66 28 B7 F0 81 A4  8C C8 3B 3E 85 65 B1 C2  ..f(......;>.e..
000005b0  6B 81 BE E6 E1 7C D3 13  26 57 25 79 7B E5 A2 5F  k....|..&W%y{.._
000005c0  C7 88 87 7F 7A 88 A4 07  D0 5F C4 D5 C4 76 98 AF  ....z...._...v..
```

5. Bonus:

   a) Main modifications based on the previous program:

      i. Extra information in FCB:

      Based on the previous FCB, it uses some extra bytes to store the information:

1) The 28th byte is used to indicate the entry type. If it equals to 0xff means it is a file, otherwise it will be a folder, and the byte will record the number of files in the folder.
2) The 29th and 30th bytes will used to be the identifier, which is a pointer of the FCB of the folder contains the current file.

ii. Global variables:

1) int level_2:

   Record the second level directory, if it equals to -1 means it is in the root folder.
2) int level_3:

   Record the third level directory, if it equals to -1 means it is in the root folder or a second level folder.

iii. void fs_gsys(FileSystem *fs, int op):

1) if op equals to LS_D or LS_S:

   In this program, the system will also go through all the FCB entries like the basic part. However, it will only print out the files in the current directory.
2) If op equals to CD_P:

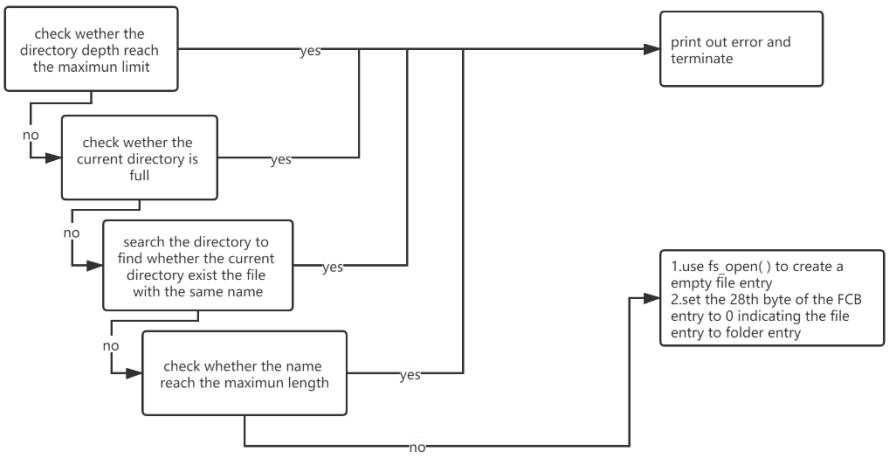
   If will change the level_2 and level_3. That is, if the level_3 is not -1, it will set it to -1. Otherwise, it will check level_2, if level_2 is not -1, it will set level_2 to -1.
3) If op equals to PWD:

   It will display the current folder based on the level_2 and level_3.

iv. void fs_gsys(FileSystem *fs, int op, char *s):

1) if op equals to MKDIR:

   diagram of the basic logic

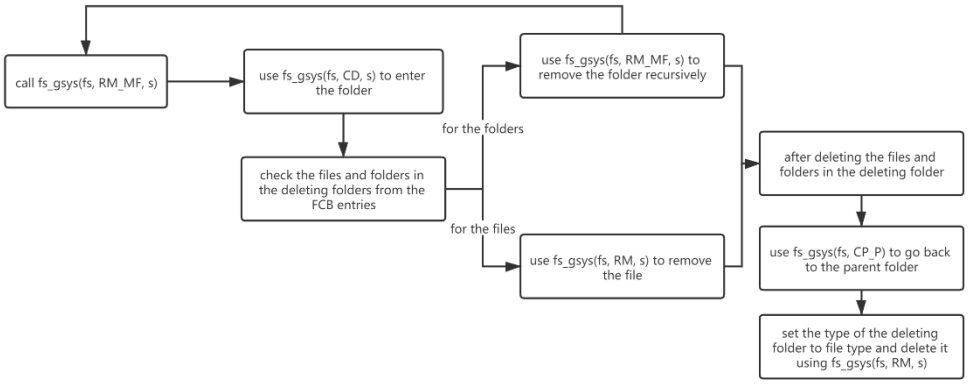


2) if op equals to CD:

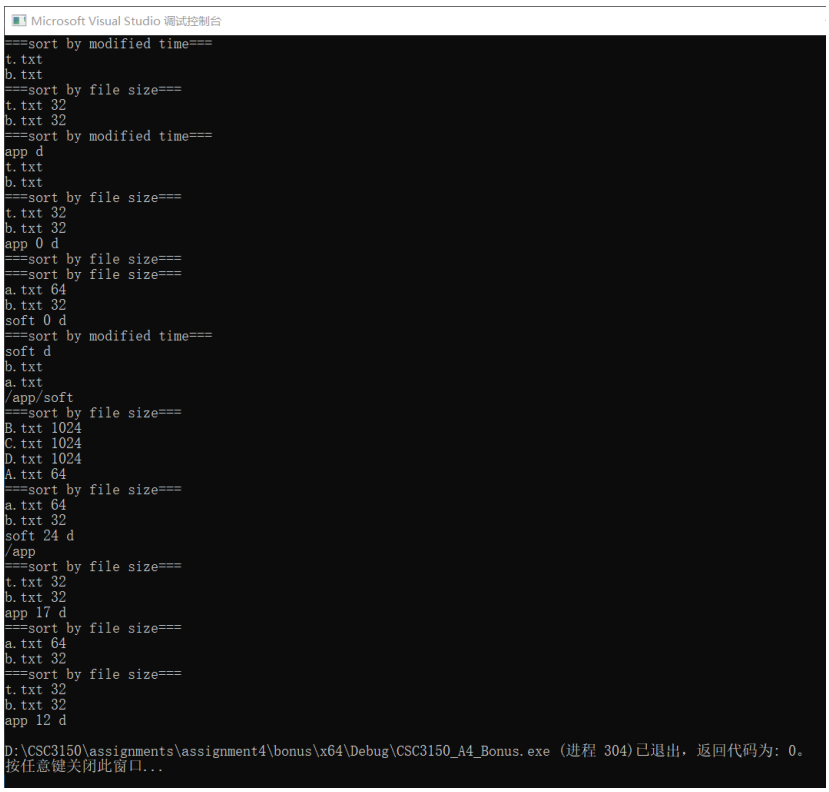
First, the program will check whether the folder exists in the current folder. If so, it will change the level_2 and level_3 accordingly. If the level_2 is -1 then it will set level_2 to the pointer of the FCB entry of the folder. Otherwise, it will set level_3 to the pointer of the FCB entry of the folder.

3) if op equals to RM_MF:

diagram of the basic logic



b) outputs:



```
Microsoft Visual Studio 调试控制台
===sort by modified time===
t.txt
b.txt
===sort by file size===
t.txt 32
b.txt 32
===sort by modified time===
app d
t.txt
b.txt
===sort by file size===
t.txt 32
b.txt 32
app 0 d
===sort by file size===
===sort by file size===
a.txt 64
b.txt 32
soft 0 d
===sort by modified time===
soft d
b.txt
a.txt
/app/soft
===sort by file size===
B.txt 1024
C.txt 1024
D.txt 1024
A.txt 64
===sort by file size===
a.txt 64
b.txt 32
soft 24 d
/app
===sort by file size===
t.txt 32
b.txt 32
app 17 d
===sort by file size===
a.txt 64
b.txt 32
===sort by file size===
t.txt 32
b.txt 32
app 12 d

D:\CSC3150\assignments\assignment4\bonus\x64\Debug\CSC3150_A4_Bonus.exe (进程 304)已退出，返回代码为: 0。
按任意键关闭此窗口...
```

6. Problems met in the project

   When implementing this project, I met a very strange bug: I could not run any recursive function in my program, even if I did not call it, the program could not run properly, that is, there will be no output in the terminal. After checking the codes and searching on the Internet, I found this bug: previously in a function, I allocated an array which is of type uchar[1024] for temporary used. Once I deleted that array, my function would just run properly. The reason behind this I thought was that, the stack size for the cuda is not big enough to hold this large array. So I just deleted this array and used another way which avoided using extra array to implement the function.

7. What I learn from the program:

   Through this program, I learnt the basic structure of the file system. Also, by implementing the simple file system operations, I also learnt about how to do the memory compaction and set superblock bits. In the bonus part, I learnt how to design a simple directory system like how to represent a directory by a file. By

implementing the operations like MKDIR and RM_MF, I learnt some basic knowledge about the folder operations like create a folder and remove a folder recursively.