THE CNINESE UNIVERSITY OF HONG KONG, SHENZHEN

CSC3150

Operating Systems

Report for Assignment 4

Author:
Li Jingyu 李璟瑜

Student Number:
118010141

Nov 22, 2020

Contents

1 Design

Assignment 4 requires us to design a program that simulates the mechanism of file system. File system is common and necessary in a computer. It manages the files to make user better access and operate them.

The whole structure of the project is shown in the following graph. From host CPU we launch CUDA Kernel, doing the I/O operations to load data. In CUDA GPU, input buffer and output buffer are used to test the file system. They are the source and the terminal of the data. Temporary space is used to store the temporary data/variables created, which does not have much existence in the project. The most important part is the file system (volume), which include super control block (I), file control block (II) and file contents (III). For each function, we need to deal with these three parts properly.

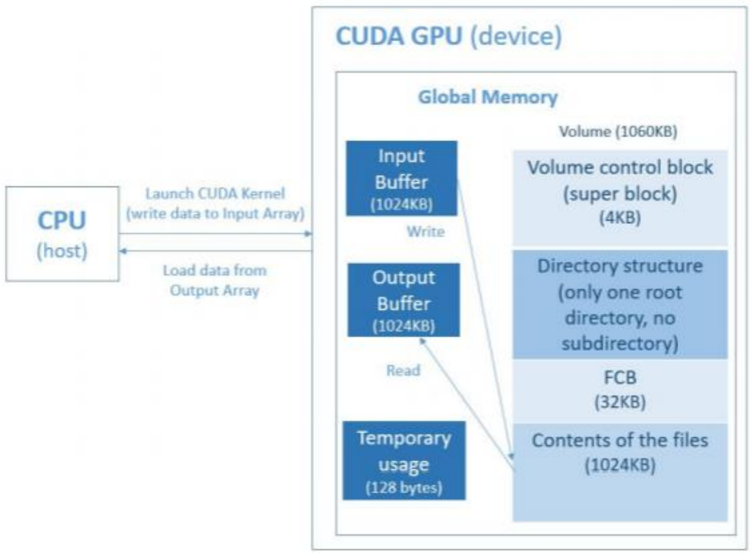

CUDA GPU (device)
Global Memory
Volume (1060KB)
Input Buffer (1024KB)
Write
Output Buffer (1024KB)
Read
Temporary usage (128 bytes)
Volume control block (super block) (4KB)
Directory structure (only one root directory, no subdirectory)
FCB (32KB)
Contents of the files (1024KB)
CPU (host)
Launch CUDA Kernel (write data to Input Array)
Load data from Output Array

Figure 1: Project structure

I. Super control block

Super control block is used to record the state of the disk memory. It is just like a free space manager. If a space is available for storing a file (a block), the state is 0, or it is 1 if it has already been occupied. Super block implements bit-map mechanism, which means one bit represents one block. The unit for accessing the super block is uchar (8 bytes), and it can represent 8 blocks/bits. So the translation between the bytes and bits is needed. Here I use the shift operation and addition & subtraction to translate the relationship. The total size is 4KB for super control block and there are in total 4096 bytes and 32768 bits to represent 32768 blocks.

II. File control block

FCB is used to record all the information of a file except its real contents. The FCB I design in the project includes the following parts:

a. File name. It occupies 20 bytes with unit uchar (unsigned char). It is the largest since a file name can be a very long string. When one wants to access to file name, he had better use %s and the pointer pointing to the first character.

b. Modify time. It occupies 2 bytes with the unit short. It represents the time when modifying a file (write a file). It is a factor used to sort the file.

c. Create time. It occupies 2 bytes with the unit short. It represents the time when creating a file (open a non-existing file). It is a factor used to sort the file when two files have the same size.

d. File size. It occupies 4 bytes with the unit u32 (unsigned int). It represents how many **bytes** a file occupies (not block). It is a factor used to sort the file.

e. File location. It occupies 4 bytes with the unit u32. It is used to record the block location of the file. Pay attention that the unit is the block instead of the byte. When knowing the location, one can access the real file content and do the read or write operations.

In summary, the FCB consists of file name + modify time + create time + size + location (20+2+2+4+4=32). The 32 bytes space is used up. There are 1024 entries (maximum file number: 1024) so the total size is 32KB for FCB.

III. File contents

The file contents part include the real file content. Its basic unit is uchar, which is 1 byte. There are in total 1024KB to store all the file contents, which is a relatively huge storage space in this project. When doing the read/write operation, one needs to access the content byte by byte. And one needs to notice that the minimum unit of a file is 32 bytes (1 block) so there is internal fragmentation (a file does not occupy the whole 32 bytes space). The file content part should be viewed as the partition of many blocks.

Next I will talk about my design for the five specific functions we need to code in the project, following the program logic flow.

I. The detailed process for fs_open():

1. Check the op state. If it is READ, set fp as 0. If it is WRITE set fp as 1024. They are used to distinguish two modes.
2. Iterate the FCB to find the file name. If it is found, return fp, which includes its location (index i). If not found, go to step 3.
3. If it is WRITE mode, create a new zero byte. Fill the FCB information using the file name and the gtime (create time), set the file size 0.
4. Find an empty bit in the super control block, doing the bit-wise operation. Put the entry to fp and return it. Now fp includes two information: read/write mode and the corresponding fcb entry location.

II. The detailed process for fs_read():

1. Obtain the mode and fcb entry from fp.
2. Access the corresponding file information from fcb, locate the file content location and read the content byte by byte.

III. The detailed process for fs_write():

1. Obtain the mode and fcb entry from fp.
2. Access the file information and calculate how many bits the new and the original files occupy.
3. Case 1: If writing a new file into an empty space, update the fcb size, then write the content inside byte by byte. Update the bit-map (add some bits).

4. Case 2: If there is an old file, first traverse the FCB to find which files need to be compacted. Record their index and use an array to store it. Then do the bubble sort according to their location (Just change the index). Then according to the location order, move the file contents from small location to large one (avoid covering). Then update bit-map, here only the bit difference needs to be considered. At last, update the location, size of the new file and write the contents to the file content part.

IV. The detailed process for fs_gsys(RM):

The process for delete is pretty similar with the write operation. Also, traverse the FCB and find the file to be deleted. Compact all the files behind. Bubble sort their locations. Update the file content. Update the bit-map (eliminate some bits). Clear the original FCB information.

V. The detailed process for fs_gsys(LS_D / LS_S):

1. Iterate the FCB entry, find non-empty entry and store its index into an array.
2. Sort the file according modified time or size, using bubble sort, after each comparison, relative file index will be moved. For the size sort, if two files have the same size, then first create bigger.
3. Print out the information according to the sorted fcb index (extract information from it).

Some reflections:

1. Since the program asks to implement continuous allocation, the file will be compacted after every operation that needs to move the file position. There will never be external fragmentation in this case. As a result, the bit-map will always fill the bits at the front and only the first available bit in the bit-map needs to be recorded. Every time one just needs to modify its location (just like moving a pointer). I have set a global variable representing the empty entry. It is for debugging. Anyway I still implement the bit-wise operation on the super block, doing the troublesome calculation.
2. The volume needs to be initialized to 0/'\0' at first in main function (at least I do), because the initial value in a uchar array is not zero. When I test it using volume[0]==0 it prints false.
3. Continuous allocation is relatively simple among other allocation algorithms. However, the efficiency is low because every time all the files behind the targeted file needs to be moved, with complexity O(N).
4. Bubble sort is also simple to implement but low in efficiency. It takes O(N^2) to sort and do the swapping.
5. The method I use to calculate the number of bits a file occupy using its size is a little bit tricky. Since when a file is 1-32 bytes, it occupies 1 bit. The sentence I use is: (using the ceil() function to get the smallest larger integer)

```
u32 bit_num_new = ceil(double(size)/fs->STORAGE_BLOCK_SIZE);
```

6. To judge whether a file exists in a fcb entry, I just check whether the first character is '\0'. Also one can check whether the file size is 0.

7. Write operation is the most complex and troublesome operation to implement. Many details need to be noticed. Continuous allocation is used that if there is an old file there, then delete it, compact all the following files and write the new file at the end. The unit to compact is byte. Delete is rather similar with write operation and other operations are relatively simple.

For more details, please refer to the codes.

2 Problems and Solutions

In this part I will discuss many problems (bugs) I met and how I solved them when writing the assignment.

(a) **Problem**: Understanding of the file system.

Solution: At first it is hard for me to make the mechanism clear. I am confused about the terminology. Go to the lecture and the tutorial. Discuss with peers s. Use Internet to learn more knowledge about the file system.

(b) **Problem**: Sort strategy.

Solution: At first I want to make use of the sort() function in <algorithm> library to simplify the process. However, I find it incompatible with CUDA and it is hard to implement structure and std::vector and allocate the memory. Then I give up and use bubble sort.

(c) **Problem**: Lethal Typo.

Solution: I write a variable into another variable (len->i), which costs me one hour to debug.

(d) **Problem**: Unit confusion.

Solution: At first I always cannot make the difference between the size and the block clear. I think that location is with the unit bytes and debug for a large amount of time! This is basic definition confusion. Size should be bytes and location should be with blocks.

(e) **Problem**: Bit-map implementation.

Solution: I wrongly add the offset of the bit-map and spent a long time drawing the numbers and deal with the troublesome bit-byte translation. Finally make it clear and correct the bug.

(f) **Problem**: Stupid surplus.

Solution: I add a break after the iteration and forget that there can be empty fcb entry between two existing entries.

(g) **Problem**: Pointer operation.

Solution: I often forget to add the * to access to contents the pointer points to. Instead I directly use the pointer itself, but it represents an address!

3 Execution

OS : **Win 10**

VS version: **Visual Studio 2015 (v140)**

CUDA version: **9.2**

GPU information: **NVIDIA Geforce GTX 1060**

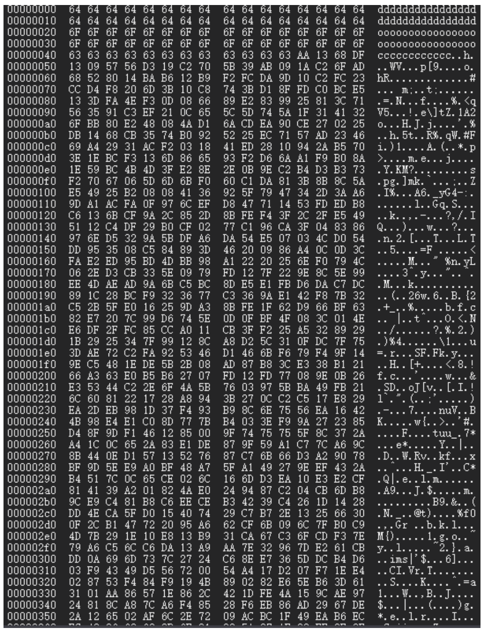
Open Visual Studio, load the project (.sln) file, use Ctrl+F7 to compile all the .cu files, and then use ctrl+F5 to run the program.

Relative program outputs will show in the command line window. The running time of this assignment is shorter than the previous one (less than 1 minute).

4 Output

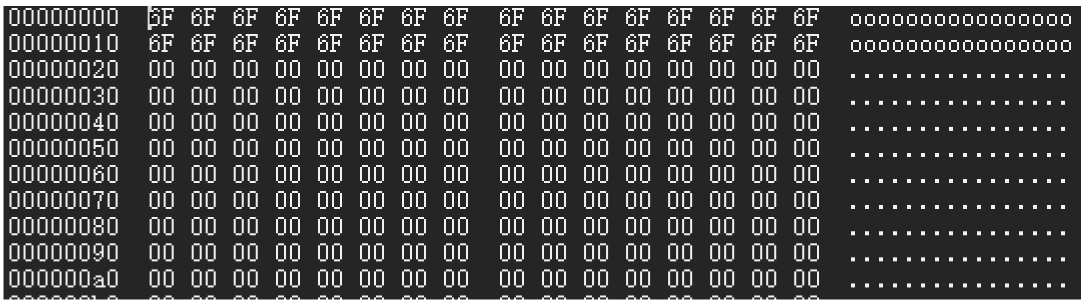
In this part the relevant program outputs will be shown.

a. bin



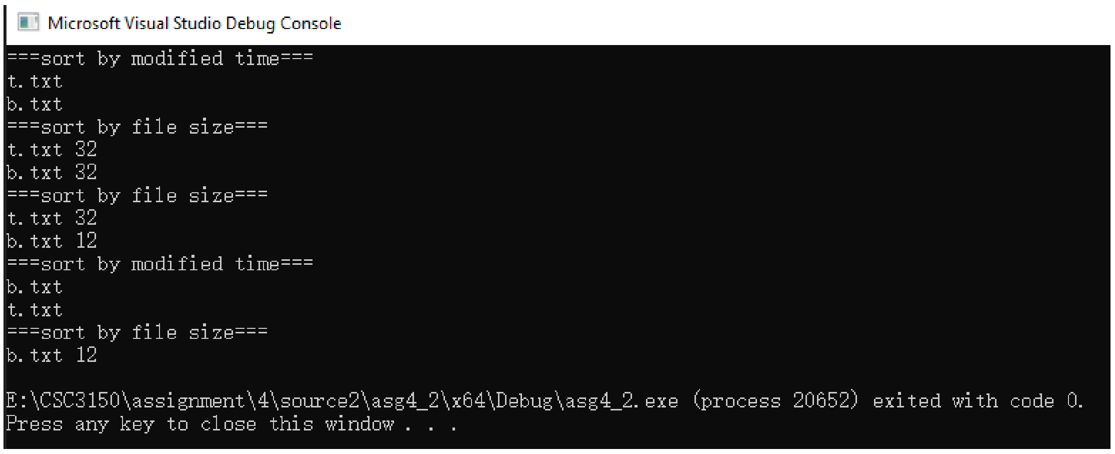
```
00000000  64 64 64 64 64 64 64 64  64 64 64 64 64 64 64 64  dddddddddddddddd
00000010  64 64 64 64 64 64 64 64  64 64 64 64 64 64 64 64  dddddddddddddddd
00000020  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000030  6F 6F 6F 6F 6F 6F 6F 6F  6F 6F 6F 6F 6F 6F 6F 6F  oooooooooooooooo
00000040  63 63 63 63 63 63 63 63  63 63 63 63 AA 13 68 DF  cccccccccccc..h.
00000050  13 09 57 56 D3 19 C2 70  5B 39 AB 09 1A C2 6F AD  ..WV...p[9....o.
00000060  68 52 80 14 BA B6 12 B9  F2 FC DA 9D 10 C2 FC 23  hR.............#
00000070  CC D4 F8 20 6D 3B 10 C8  74 3B D1 8F FD C0 BC E5  ... m;..t;......
00000080  13 3D FA 4E F3 0D 08 66  89 E2 83 99 25 81 3C 71  .=.N...f....%.<q
00000090  56 35 91 C3 EF 21 0C 65  5C 5D 74 5A 1F 31 41 32  V5...!.e\]tZ.1A2
000000a0  6F BB 80 E2 48 08 4A D1  6A CD EA 90 CE 27 02 25  o...H.J.j....'.%
000000b0  DB 14 68 CB 35 74 B0 92  52 25 EC 71 57 AD 23 46  ..h.5t..R%.qW.#F
000000c0  69 A4 29 31 AC F2 03 18  41 ED 28 10 94 2A B5 70  i.)1....A.(..*.p
000000d0  3E 1E BC F3 13 6D 86 65  93 F2 D6 6A A1 F9 B0 8A  >....m.e...j....
000000e0  1E 59 BC 4B 4D 3F E2 8E  2E 0B 9E C2 B4 D3 B3 73  .Y.KM?.........s
000000f0  F2 70 67 06 5D 6D 6B F0  60 C1 DA 81 3B 8B 8C 5A  .pg.]mk.`...;..Z
00000100  E5 49 25 B2 08 08 41 36  92 5F 79 47 34 2D 3A A6  .I%...A6._yG4-:.
00000110  9D A1 AC FA 0F 97 6C EF  D8 47 71 14 53 FD ED B8  ......l..Gq.S...
00000120  C6 13 6B CF 9A 2C 85 2D  8B FE F4 3F 2C 2F E5 49  ..k..,.-...?,/.I
00000130  51 12 C4 DF 29 B0 CF 02  77 C1 96 CA 3F 04 83 86  Q...)...w...?...
00000140  97 6E D5 32 9A 5B DF A6  DA 54 E5 07 03 4C D0 54  .n.2.[...T...L.T
00000150  DD 95 35 08 C5 84 89 3D  46 20 09 86 A4 0C 0D 3C  ..5....=F .....<
00000160  FA E2 ED 95 BD 4D BB 98  A1 22 20 25 6E F0 79 4C  .....M..." %n.yL
00000170  06 2E D3 CB 33 5E 09 79  FD 12 7F 22 9E 8C 5E 99  ....3^.y..."..^.
00000180  EE 4D AE AD 9A 6B C5 BC  8D E5 E1 FB D6 DA C7 DC  .M...k..........
00000190  89 1C 28 BC F9 32 36 77  C3 36 9A E1 42 F8 7B 32  ..(..26w.6..B.{2
000001a0  C5 2B 5F E0 16 25 9D A3  8B FE 1F 62 D9 66 BF 63  .+_..%.....b.f.c
000001b0  82 E7 20 7C 99 D6 74 5E  0D 0F BF 4F 08 3C 01 4E  .. |..t^...O.<.N
000001c0  E6 DF 2F FC 85 CC A0 11  CB 3F F2 25 A5 32 89 29  ../......?.%.2.)
000001d0  1B 29 25 34 7F 99 12 8C  A8 D2 5C 31 0F DC 7F 75  .)%4......\1...u
000001e0  3D AE 72 C2 FA 92 53 46  D1 46 6B F6 79 F4 9F 14  =.r...SF.Fk.y...
000001f0  9E C5 48 1E DE 5B 2B 08  AD 87 B8 3C E3 38 B1 21  ..H..[+....<.8.!
00000200  66 A3 63 E0 B5 B6 27 07  FD 12 FD 77 08 9E 0B 26  f.c...'....w...&
00000210  E3 53 44 C2 2E 6F 4A 5B  76 03 97 5B BA 49 FB 21  .SD..oJ[v..[.I.!
00000220  6C 60 81 22 17 28 A8 94  3B 27 0C C2 C5 17 E8 29  l`.".(..;'.....)
00000230  EA 2D EB 98 1D 37 F4 93  B9 8C 6E 75 56 EA 16 42  .-...7....nuV..B
00000240  4B 98 E4 E1 C0 8D 77 7B  B4 03 3E F9 9A 27 23 85  K.....w{..>..'#.
00000250  D4 8F 9D F1 46 12 85 00  9F 74 75 75 5F 8C 37 2A  ....F....tuu_.7*
00000260  A4 1C 0C 65 2A 83 E1 DE  87 9F 59 A1 C7 7C A6 9C  ...e*.....Y..|..
00000270  8B 44 0E D1 57 13 52 76  87 C7 6B 66 D3 A2 90 78  .D..W.Rv..kf...x
00000280  BF 9D 5E E9 A0 BF 48 A7  5F A1 49 27 9E EF 43 2A  ..^...H._.I'..C*
00000290  B4 51 7C 0C 65 CE 02 6C  16 6D D3 EA 10 E3 E2 CF  .Q|.e..l.m......
000002a0  81 41 39 A2 01 82 4A E0  24 94 87 C2 04 CB 6D B8  .A9...J.$.....m.
000002b0  9C E9 C4 81 B8 C6 EE CE  B3 42 39 C4 26 1D 14 28  .........B9.&..(
000002c0  DD 4E CA 5F D0 15 40 74  29 C7 B7 2E 13 25 66 30  .N._..@t)....%f0
000002d0  0F 2C B1 47 72 20 95 A6  62 CF 6B 09 6C 7F B0 C9  .,.Gr ..b.k.l...
000002e0  4D 7B 29 1E 10 E8 13 B9  31 CA 67 C3 6F CD F3 7E  M{).....1.g.o..~
000002f0  79 A6 C5 6C C6 DA 13 A9  AA 7E 32 96 7D E2 61 CB  y..l.....~2.}.a.
00000300  DD 0A 69 6D 73 7C 27 24  C6 8E E7 36 5D DC B4 D6  ..ims|'$...6]...
00000310  03 F9 43 49 D5 56 72 00  54 A4 17 D2 07 F7 1E E4  ..CI.Vr.T.......
00000320  02 87 53 F4 84 F9 19 4B  89 02 82 E6 5E B6 3D 61  ..S....K....^.=a
00000330  31 01 AA 86 57 1E 86 2C  42 1D FE 4A 15 9C AE 97  1...W..,B..J....
00000340  24 81 8C A8 7C A6 F4 85  28 F6 EB 86 AD 29 67 DE  $...|...(....)g.
00000350  2A 12 65 02 AF 6C 2E 72  09 AC BC 1F 49 EA B6 EC  *.e..l.r....I...
```

b. snapshot



```
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  ................
```

c. test case 1

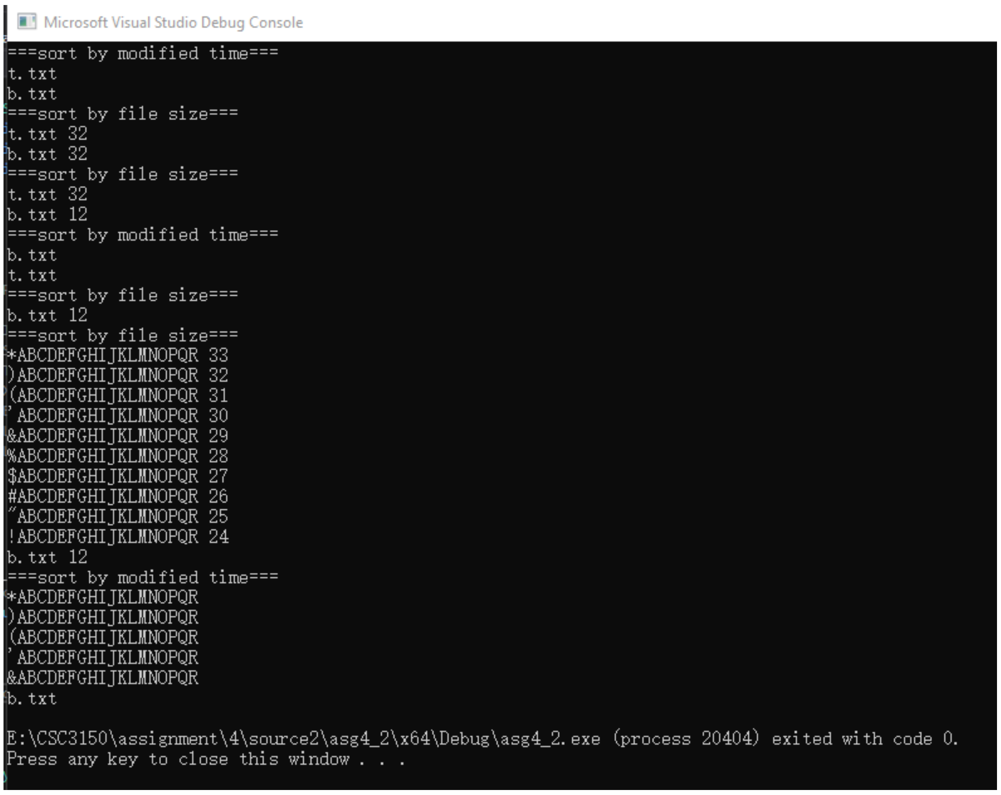


Microsoft Visual Studio Debug Console

```
===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

E:\CSC3150\assignment\4\source2\asg4_2\x64\Debug\asg4_2.exe (process 20652) exited with code 0.
Press any key to close this window . . .
```

d. test case 2

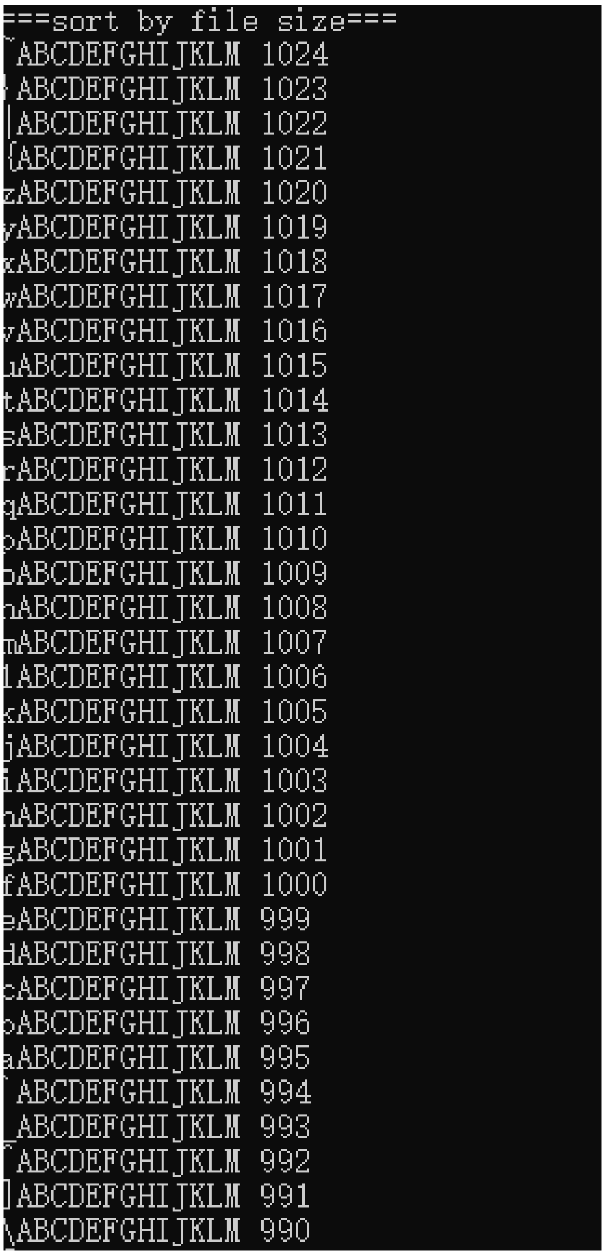


Microsoft Visual Studio Debug Console

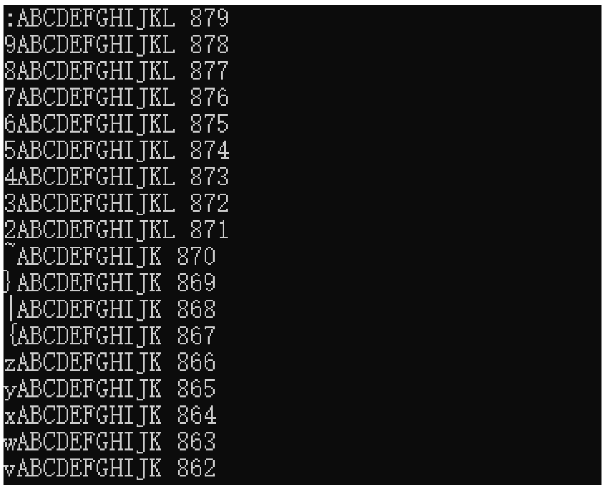
```
===sort by modified time===
t.txt
b.txt
===sort by file size===
t.txt 32
b.txt 32
===sort by file size===
t.txt 32
b.txt 12
===sort by modified time===
b.txt
t.txt
===sort by file size===
b.txt 12
===sort by file size===
*ABCDEFGHIJKLMNOPQR 33
)ABCDEFGHIJKLMNOPQR 32
(ABCDEFGHIJKLMNOPQR 31
'ABCDEFGHIJKLMNOPQR 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

E:\CSC3150\assignment\4\source2\asg4_2\x64\Debug\asg4_2.exe (process 20404) exited with code 0.
Press any key to close this window . . .
```

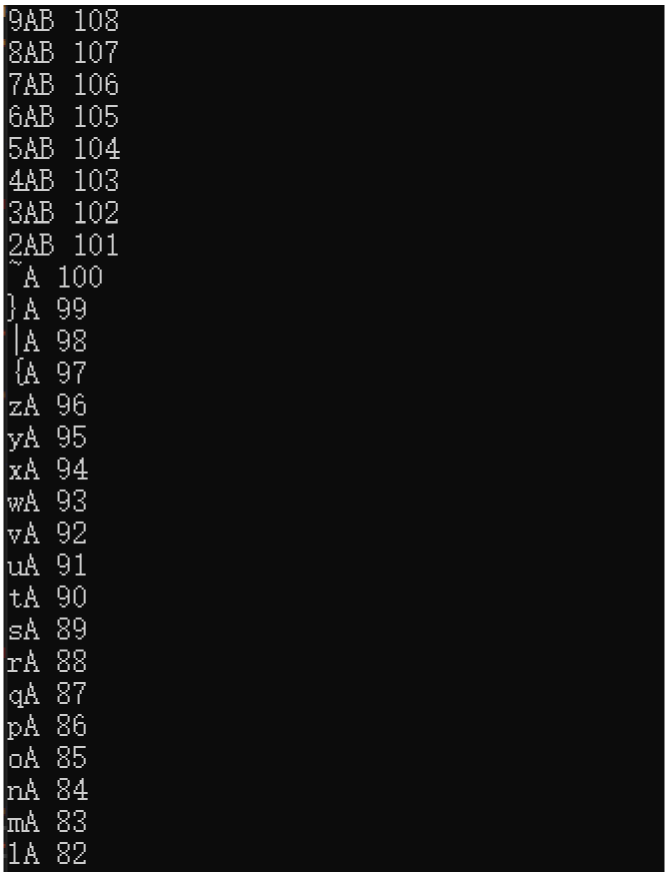
e. test case 3 (part)



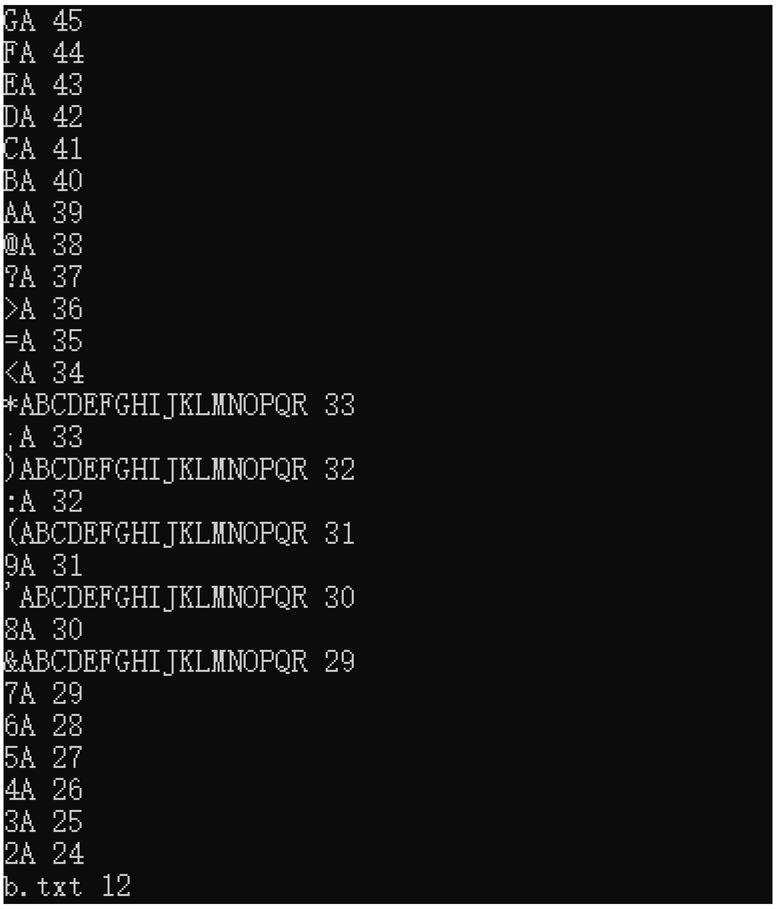
```
===sort by file size===
~ABCDEFGHIJKLM 1024
}ABCDEFGHIJKLM 1023
|ABCDEFGHIJKLM 1022
{ABCDEFGHIJKLM 1021
zABCDEFGHIJKLM 1020
yABCDEFGHIJKLM 1019
xABCDEFGHIJKLM 1018
wABCDEFGHIJKLM 1017
vABCDEFGHIJKLM 1016
uABCDEFGHIJKLM 1015
tABCDEFGHIJKLM 1014
sABCDEFGHIJKLM 1013
rABCDEFGHIJKLM 1012
qABCDEFGHIJKLM 1011
pABCDEFGHIJKLM 1010
oABCDEFGHIJKLM 1009
nABCDEFGHIJKLM 1008
mABCDEFGHIJKLM 1007
lABCDEFGHIJKLM 1006
kABCDEFGHIJKLM 1005
jABCDEFGHIJKLM 1004
iABCDEFGHIJKLM 1003
hABCDEFGHIJKLM 1002
gABCDEFGHIJKLM 1001
fABCDEFGHIJKLM 1000
eABCDEFGHIJKLM 999
dABCDEFGHIJKLM 998
cABCDEFGHIJKLM 997
bABCDEFGHIJKLM 996
aABCDEFGHIJKLM 995
`ABCDEFGHIJKLM 994
_ABCDEFGHIJKLM 993
^ABCDEFGHIJKLM 992
]ABCDEFGHIJKLM 991
\ABCDEFGHIJKLM 990
```



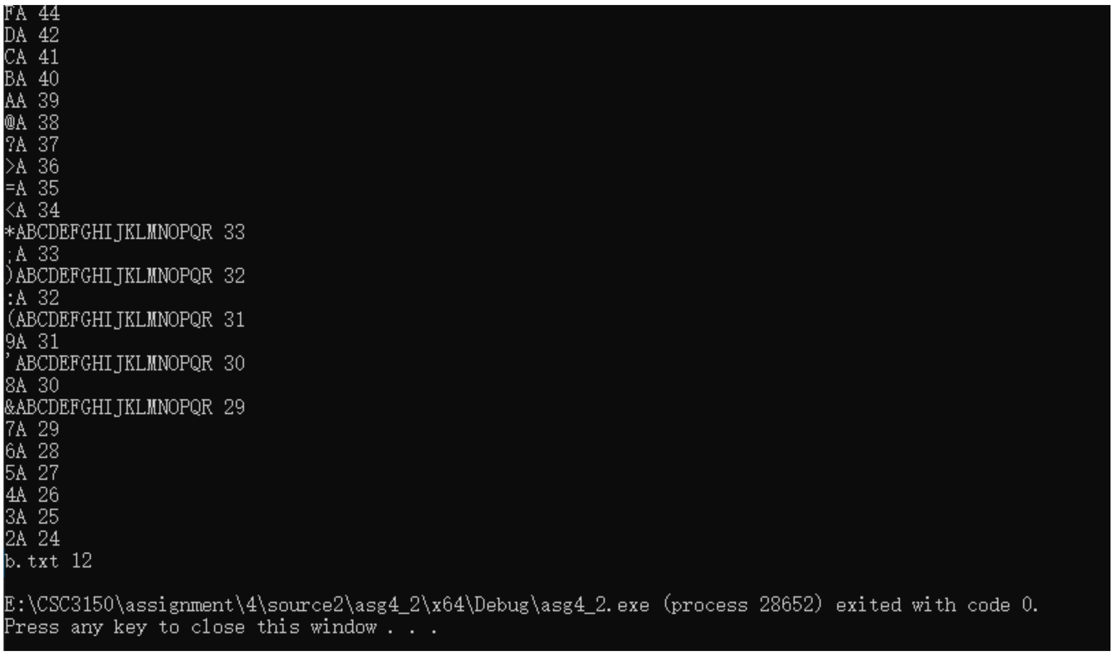
```
:ABCDEFGHIJKL 879
9ABCDEFGHIJKL 878
8ABCDEFGHIJKL 877
7ABCDEFGHIJKL 876
6ABCDEFGHIJKL 875
5ABCDEFGHIJKL 874
4ABCDEFGHIJKL 873
3ABCDEFGHIJKL 872
2ABCDEFGHIJKL 871
~ABCDEFGHIJK 870
}ABCDEFGHIJK 869
|ABCDEFGHIJK 868
{ABCDEFGHIJK 867
zABCDEFGHIJK 866
yABCDEFGHIJK 865
xABCDEFGHIJK 864
wABCDEFGHIJK 863
vABCDEFGHIJK 862
```



```
9AB 108
8AB 107
7AB 106
6AB 105
5AB 104
4AB 103
3AB 102
2AB 101
~A 100
}A 99
|A 98
{A 97
zA 96
yA 95
xA 94
wA 93
vA 92
uA 91
tA 90
sA 89
rA 88
qA 87
pA 86
oA 85
nA 84
mA 83
lA 82
```



```
GA 45
FA 44
EA 43
DA 42
CA 41
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
```



```
FA 44
DA 42
CA 41
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

E:\CSC3150\assignment\4\source2\asg4_2\x64\Debug\asg4_2.exe (process 28652) exited with code 0.
Press any key to close this window . . .
```

5 Feeling

Here I will share several feelings I have when writing assignment 4.

a. Assignment 4 is similar with assignment 3. I need to spend a lot of time understanding the basic concept, discussing with classmates to make the logic clear. However, I still not figure out some basic definitions clearly (such as the block and the byte) and it brings me a lot of troubles when debugging. So I think I must make all the things pretty clear and then start to write the code next time.

b. After discussion, we all think that this project is as straightforward as the assignment 3. At least the logic is simple and clear. However, TA Hong said we need to spend a lot of time on it and the workload is large. Writing hundreds lines of codes is needed. It proves that he is correct and I pay the price for looking down on it. I spend more time than before when coding and debugging.

c. Writing in TC301: This time we still need to go to the computer room to use the GPU to run the program. It is magical that there are not so many people writing the codes in the room close to the deadline. I feel comfortable sitting in the empty room, debugging my code (also anxious!!). Maybe many students just stay up late finishing the coding? I think I have to test the program in the room in advance.

d. The project is pretty helpful and meaningful. It deepens my understanding of the file system and relative operating system concept. I also improve my programming ability especially the usage of the pointers quite a lot. I just feel like I am quite familiar with the pointers now (let me think of the project 2 of CSC 3050, also many about the pointer usage). Great homework.

e. I think I can use more functions to encapsulate the procedure next time.

f. I do not expect that I will spend quite a lot of time in debugging this time. I watch my codes line by line, running the program again and again, thinking of the mistakes. It makes me quite painful and I find lots of bugs. Some are tiny bugs like typo and some are bugs of understanding. I just feel that I am stupid and incapable when fixing the bugs. I should go to the computer room earlier.

g. This time I try to take a shortcut using the sort() and vector instead of implementing the bubble sort hand by hand. However, it fails. I had to start from the beginning and modify the codes, which costs me even more time. I decide to walk stably next time. Little cleverness is not suitable for me. More time should be spent on this project (Haven't expected that it will be more difficult than the previous three projects).

h. It is a pity that I cannot finish the bonus because of the mistakes and bugs and the busy schedule. I feel a little bit regretful!!

i. By the way, TA is helpful in answering the questions in the tutorial, good.

That's all.