CSC3150 Operating System

Assignment Report #4

Name: Yang Yin

Student ID: 120090516

Date: 2022.11.30

The Chinese University of Hong Kong, Shenzhen

1. Design (Definition of variables)

   fs->volume space allocation:

   (1) SUPERBLOCK: first 4KB (superblock size)

   (2) FCB: 32KB after SUPERBLOCK

   A maximum of 1k files can be collected, each file information has 32B

   1) 0-19 B stored name
   2) 20-21B stored create time
   3) 22-23B stored modified time
   4) 24-25B stored address (of super block number)
   5) 26-27B stored size (The units are B)

      Specially, in bonus, for directory it stored all of the file name length under the directory.

   6) 28B stored valid bit (ff means invalid, 00 means valid, 01 means in sort)

      Specially, in bonus, f0 means valid as directory, x1means under the now directory (when sorting), x2 means has sort (when sorting).

   7) 29-30B stored parent point in bonus, when its parent is root or no parent, these bits should be 0xff

   (3) Contains of file: 1024K for total. Stored files with block is 32B.

   gtime: gtime is increased every time a file (or a directory) is updated

   gsize: gsize marks the number of blocks used.

   file_number: Records the number of existing files

   now_directory: Mark the current directory, which is 0xffff when in the root directory

2. Design (Definition of function)

   (1) FCB_init: Set the valid bit of all FCBs to 0xff

   Specially, in bonus, also set the parent bit of FCB to 0xff

   (2) SUPERBLOCK_init: set all of superblock to zero.

   (3) string_print: print string with pointer s.

   (4) check_name: check strings s1 and s2 same or not

   (5) update_parent_size: Change the size of the directory when the file name changes

(6) update_file_name: Try writing s as the name of the fp file to check if the name is too long

(7) Modified_FCB: Update FCB of file based on size and gtime.

(8) get_len: Returns the length of the string s

(9) set_superblock_bit: Changes the superblock information for the addr position based on bit

(10) Remove_file: Clear the storage space of the file fp, and move the storage location of the following file forward

(11) Cmp: The preference of file b is true according to the judgment of file a and file b which has a higher priority in the arrangement according to op.

(12) Output_parent: Rely on recursive output file paths

(13) Others function: Definition as required

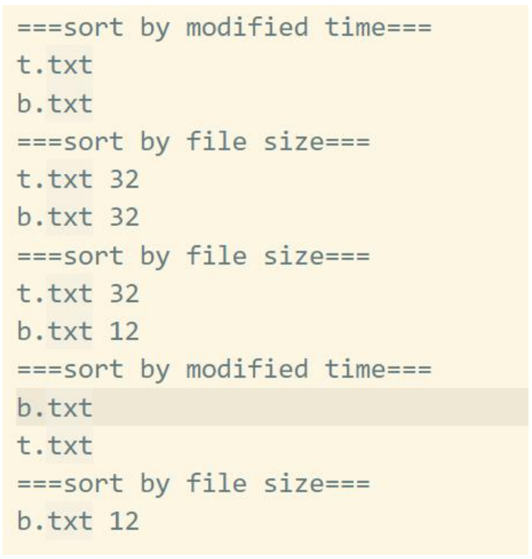
3. Environment

Use Cluster.

System Type	x86_64
Opearing System	CentOS Linux release 7.5.1804
CPU	Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz 20 Cores, 40 Threads
Memory	100GB RAM
GPU	Nvidia Quadro RTX 4000 GPU x 1
CUDA	11.7
GCC	Red Hat 7.3.1-5
CMake	3.14.1

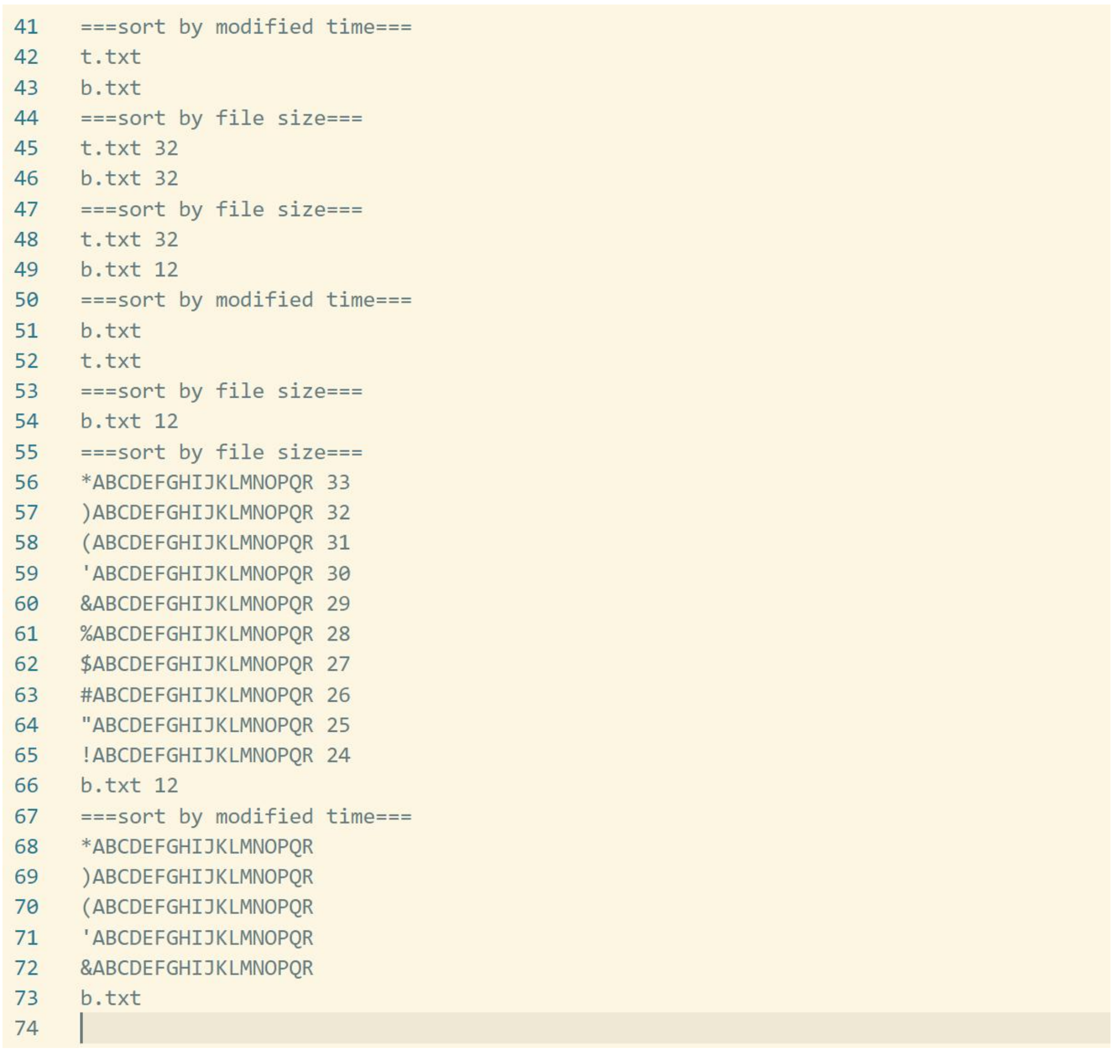
4. Screenshot

(1) Case1:



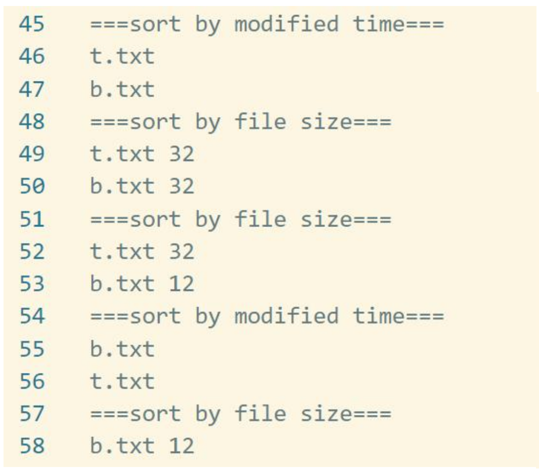
```
===sort by modified time===
t.txt
b.txt
===sort by file size===
t.txt 32
b.txt 32
===sort by file size===
t.txt 32
b.txt 12
===sort by modified time===
b.txt
t.txt
===sort by file size===
b.txt 12
```

(2) Case 2

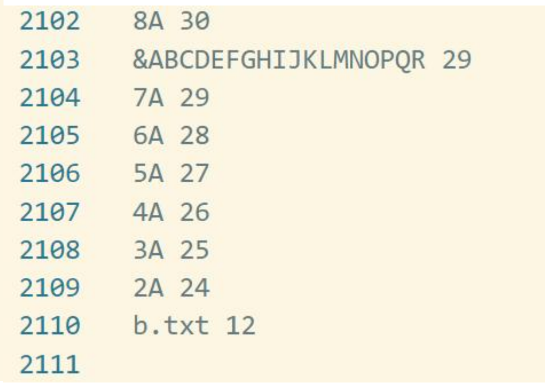


```
===sort by modified time===
t.txt
b.txt
===sort by file size===
t.txt 32
b.txt 32
===sort by file size===
t.txt 32
b.txt 12
===sort by modified time===
b.txt
t.txt
===sort by file size===
b.txt 12
===sort by file size===
*ABCDEFGHIJKLMNOPQR 33
)ABCDEFGHIJKLMNOPQR 32
(ABCDEFGHIJKLMNOPQR 31
'ABCDEFGHIJKLMNOPQR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

(3) Case 3(Fore and aft)



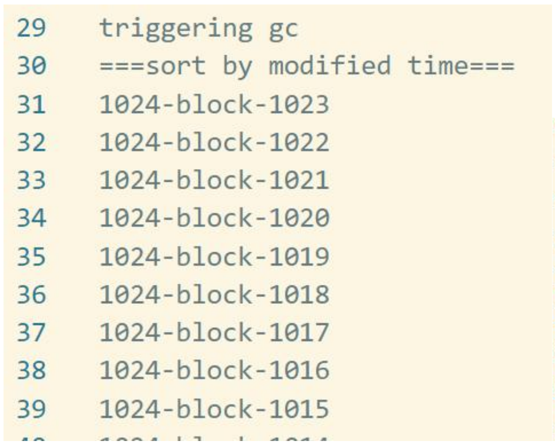
```
===sort by modified time===
t.txt
b.txt
===sort by file size===
t.txt 32
b.txt 32
===sort by file size===
t.txt 32
b.txt 12
===sort by modified time===
b.txt
t.txt
===sort by file size===
b.txt 12
```



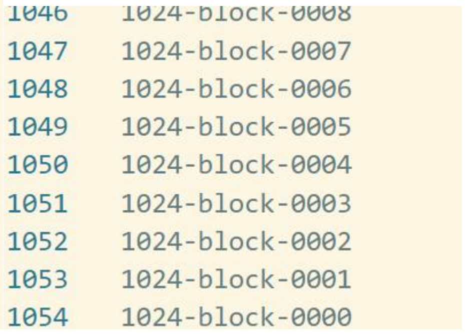
```
8A 30
&ABCDEFGHIJKLMNOPQR 29
7A 29
6A 28
5A 27
4A 26
3A 25
2A 24
b.txt 12

```

(4) Case 4 (Fore and aft)

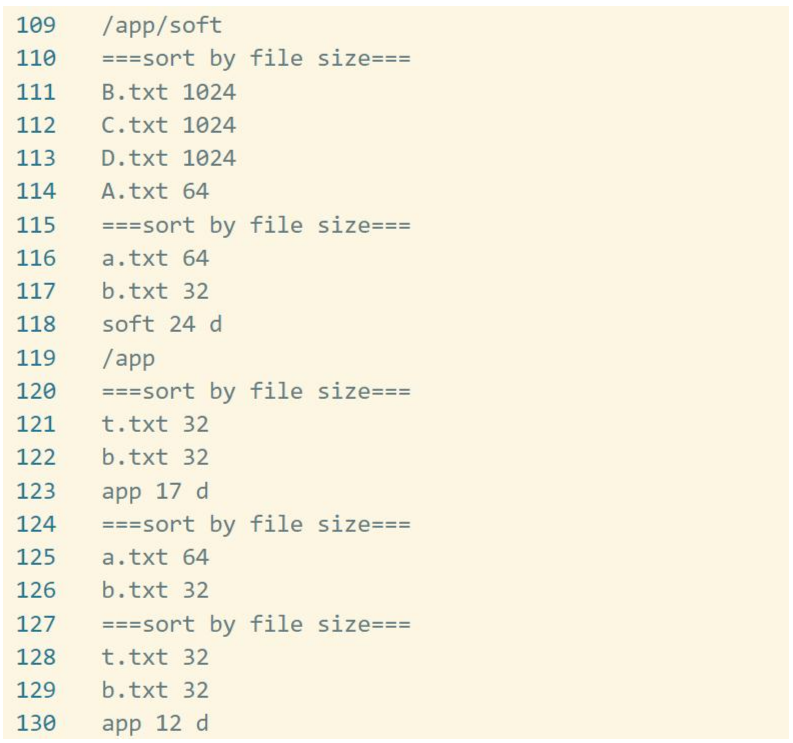


```
triggering gc
===sort by modified time===
1024-block-1023
1024-block-1022
1024-block-1021
1024-block-1020
1024-block-1019
1024-block-1018
1024-block-1017
1024-block-1016
1024-block-1015
```



```
1024-block-0008
1024-block-0007
1024-block-0006
1024-block-0005
1024-block-0004
1024-block-0003
1024-block-0002
1024-block-0001
1024-block-0000
```

(5) Case bonus (aft)



```
/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
```

5. Flow of some function

(1) fs_open

① Check whether there is a valid file named s in the FCB.

② If yes, return the serial number

③ Else, Look for an invalid FCB

④ Use update_file_name to write the file name.

⑤ Use modified_FCB to update the update time and file size in the FCB information.

⑥ gtime, file_number is increased, and the FCB is set to valid

⑦ In bonus, set the father bit as now_directory

(2) fs_read

① Verify that the file is readable. If not, output ERROR

② Use the FCB address bit to find the corresponding file location in the storage

③ Copy the size character to output

(3) fs_write

① Verify that the file is readable. If not, output ERROR

② Check whether the input file size is too large. If yes, output ERROR.

③ Check whether the original location needs to be cleared (the original size is not 0 and the new input occupies a different block size from the original occupied block size)

1) The previous size was 0: stored after the occupied block, after the address bit is occupied

2) The previous size and the current size occupy the same number of blocks: data is directly rewritten to the original address

3) Use remove_file to erase the previous storage, rewrite the data at the end, and update the address

④ After writing, update the data in FCB (size, address, modified time)

(4) fs_gsys

① PWD: Use output_parent to write out the path

② CP_D: Update now_directory with the parent bit in FCB of now_directory

③ LS_D/LS_S: Sort by marking files that need to be sorted and files that have already been sorted, relying on the cmp function to compare sizes.The valid bit is restored after the sorting is complete

In bonus, you first do a markup to filter out the files and directories in the now directory

④ RM: Locate the file in FCB, use remove_file to clear the address, and set FCB to invalid

⑤ MKDIR: Find the invaild page in FCB, write the information, and set the valid bit to 0xf0

⑥ CD: Search for a directory in FCB where the valid bit is 0xf0(is directory) and the parent node is now_directory.Update now_directory to the directory.

⑦ RM_RF: Locate the folder in the subfolder of now_directory and search for all the subfiles and subfolders of that folder. Subfiles are deleted with RM, and subfolders are recursively deleted with RM_RF instruction and subfolder name.Finally, set the folder to invalid.

The bonus content is highlighted