A2 Report

120090521

庞嘉扬

Environment

Item	Configuration / Version
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

How to run the program

1. Connect to cluster
2. Add the assignment folder and enter it
3. Type “sbatch ./slurm.sh” in the terminal

How I design my program

In the source file:

Below are some important structure and variable

FCB entry

There are 1024 FCB entries, each has 32 bytes.

1. The byte 0 represents permission, 0x10000000 means can write, 0x01000000 means can read. 0x00000000 means invalid.
2. Bytes 1-20 represent the name.
3. Bytes 21-22 represent block address. There are 32k blocks.
4. Bytes 23-26 represent size. The maximum file size is 1024k byte. Actually 3 bytes are enough. But I use 4 bytes here.
5. Byte 27-28 represent created time.
6. Byte 29-30 represent modified time.
7. Byte 31 is not used in the source file.

Bitmap

The first 4k byte in the volume. If the corresponding block is empty, the bitmap is 0; otherwise, 1.

Free_position

A global variable represents the empty block index. Write operation will make it larger, compact operation will make it smaller.

Below are some main functions need to be implemented in this project.

fs_open(FileSystem *fs, char *s, int op)

In the open process. I find if there already exists this file first.

If exists, the op is READ (read after write), then change permission 0x80 to 0xc0; if the op is WRITE, do compaction.

If not exists (open a new file). Initialize the new file and set its name and time.

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

Easy to implement, just read the content from blocks to output.

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

The same as read, reverse version.

Need to update the bitmap, size, time.

fs_gsys(FileSystem *fs, int op)

This function has two operations: LS_S and LS_D

LS_D: print the file ordered by modified time. I use a temporary array to store the valid file and use bubble sort to sort it.

LS_S: The implementation is similar to LS_D.

fs_gsys(FileSystem *fs, int op, char *s)

This function can remove the corresponding file.

I find this file first, clean it can do the compaction.

Below are some important functions in my program to support above functions.

clean(FileSystem *fs, int fp)

Used in re-writing and removing operations.

Need to do the compaction.

1. Update the bitmap: Set the first non-empty values to 1 (free_position – blocks need to be removed). Later set to 0.
2. Update the content in the blocks. Shift the content forward.
3. Update the address of the fcb corresponding to the shifted blocks.

In the bonus file:

The basic skeleton is similar to the source file.

FCB entry

As I mentioned above, 3 bytes are enough for size. Here I use 3 bytes for size. The remaining 2 bytes 30 and 31 are used to represent parent.

Current_position

This global variable is used to store the current directory.

fs_gsys(fs, CD_P);

The new operation in gsy() function. Just change the current directory to its parent.

fs_gsys(fs, CD, “app\0”);

The new operation in gsy() function. Just change the current directory to the target directory. Need to iterate through the fcb to find.

fs_gsys(fs, MKDIR, “app\0”);

Similar to open operation in the source file. Need to set the permission byte to 0x02 to represent that this is a directory.

fs, gsys(fs, PWD);

Find this file/directory’s ancestors and print them.

fs_gsys(fs, RM_RF, “app\0”);

Remove the directory and its sub directory and files. Use RM operation in the source implementation.

There are other small changes in the bonus part. Write operation will add size to its parent and change parent time. Remove operation will reduce the size of its parent.

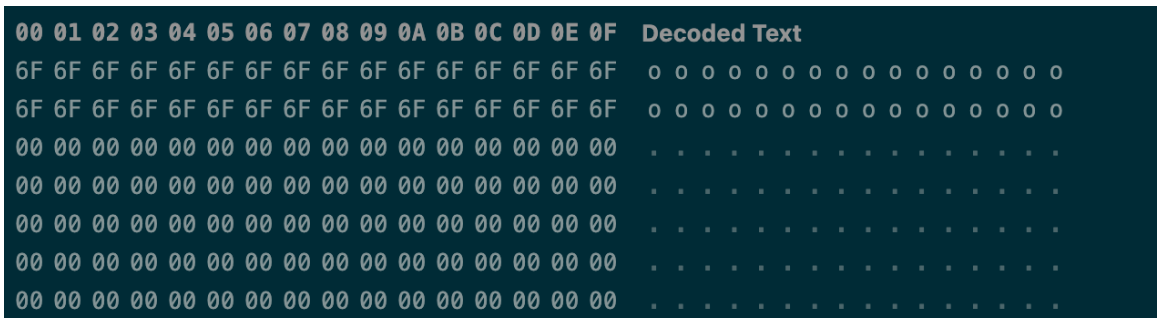
Problem

- When testing the program, I found that every time I ran it, there were different outputs. During debugging, I found that I print the file name in a function by passing a pointer. There are some problems with this method, maybe about pointer wrongs. But I still do not figure it out. Finally, I just pass the index and return the whole name and print it.
- I found that recursion would cause some problems with stack size, then I do not use it.

Screenshots

Test1

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



Test2

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

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded Text
6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F  o o o o o o o o o o o o o o o o
6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F  o o o o o o o o o o o o o o o o
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
```

Test3

```
000003B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
000003C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . . . . . . . . . . . .
000003D0  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 . ( . . . % . . . . D Z ~ . .
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  k T * . . . . . . . . I . . . .
00000420  6B 97 CA 50 8C 73 AE 66 34 07 63 D1 D1 10 3A BC  k . . P . s . f 4 . c . . . : .
00000430  64 E3 6B 51 B3 88 A4 22 1A BB 6B AA 61 9D 51 CC  d . k Q . . . " . . k . a . Q .
00000440  B5 9C 9C 42 10 4C A8 44 53 0D 95 A4 9C 50 E0 81  . . . B . L . D S . . . . P . .
00000450  B3 CB D2 67 54 F6 89 ED B2 74 98 14 92 6A 60 48  . . . g T . . . . t . . . j ` H
00000460  07 FD 8A 96 4A 33 DB 1D BF F0 41 5D C0 22 DE 75  . . . . J 3 . . . . A ] . " . u
00000470  ED 31 5C C1 28 66 AF 5A DA C7 ED EC B1 4E 35 38  . 1 \ . ( f . Z . . . . . N 5 8
00000480  CB 3F CF 95 F2 2B 32 B2 1C 73 10 DD 95 6E 53 03  . ? . . . + 2 . . s . . . n S .
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 . ? . " q y . . . . . . . ~ .
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 * _ . 3 z . . . & . .
000004D0  E9 A1 BA 21 21 DF 94 30 63 F5 1D 7A FE 33 64 FD  . . . ! ! . . 0 c . . z . 3 d .
000004E0  87 15 9F CE BE FE FA 72 F8 A3 1D 61 49 DF 68 33  . . . . . . . r . . . a I . h 3
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 + . . . . B J
00000520  64 BC 64 DD 32 6F 50 4C 98 24 AF A2 61 45 2D 41  d . d . 2 o P L . $ . . a E - A
00000530  AF 5B 25 03 5C EE B3 4F 99 42 65 0A 2C 27 54 10  . [ % . \ . . O . B e . , ' T .
00000540  E3 38 ED 17 28 3E 63 C0 E2 92 63 44 D7 90 06 88  . 8 . . ( > c . . . c D . . . .
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 . . . . . . . .
```

```
===sort by modified time===
t.txt
b.txt
===sort by size===
t.txt 32
b.txt 32
===sort by size===
t.txt 32
b.txt 12
===sort by modified time===
b.txt
t.txt
===sort by size===
b.txt 12
===sort by 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
===sort by size===
~ABCDEFGHIJKLM 1024
}ABCDEFGHIJKLM 1023
|ABCDEFGHIJKLM 1022
{ABCDEFGHIJKLM 1021
zABCDEFGHIJKLM 1020
```

Test4

```
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-1014
1024-block-1013
1024-block-1012
1024-block-1011
1024-block-1010
1024-block-1009
1024-block-1008
1024-block-1007
1024-block-1006
1024-block-1005
1024-block-1004
1024-block-1003
1024-block-1002
1024-block-1001
1024-block-1000
```


snapshot.bin
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded Text
00000000 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 d d d d d d d d d d d d d d d d
00000010 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 d d d d d d d d d d d d d d d d
00000020 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F o o o o o o o o o o o o o o o o
00000030 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F 6F o o o o o o o o o o o o o o o o
00000040 63 63 63 63 63 63 63 63 63 63 63 63 AA 13 68 DF c c c c c c c c c c c c . . h .
00000050 13 09 57 56 D3 19 C2 70 5B 39 AB 09 1A C2 6F AD . . W V . . . p [9 . . . . o .
00000060 68 52 80 14 BA B6 12 B9 F2 FC DA 9D 10 C2 FC 23 h R . . . . . #
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 V 5 . . . ! . e \ ] t Z . 1 A 2
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 . 5 t . . R % . q W . # 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 . K M ? . . . . s
000000F0 F2 70 67 06 5D 6D 6B F0 60 C1 DA 81 3B 8B 8C 5A . p g .] m k . ` . . . ; . . Z
00000100 E5 49 25 B2 08 08 41 36 92 5F 79 47 34 2D 3A A6 . I % . . . A 6 . _ y G 4 - : .
00000110 9D A1 AC FA 0F 97 6C EF D8 47 71 14 53 FD ED B8 . . . . . l . . G q . 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 . y L
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 . . (. . 2 6 w . 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 . . . S F . F k . 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 . S D . . o J [v . . [. I . !
00000220 6C 60 81 22 17 28 A8 94 3B 27 0C C2 C5 17 E8 29 l ` . " . (. . . ; ' . . . . .)

bonus

```
===sort by modified time===
t.txt
b.txt
===sort by size===
t.txt 32
b.txt 32
===sort by modified time===
app d
t.txt
b.txt
===sort by size===
t.txt 32
b.txt 32
app 0 d
===sort by size===
===sort by size===
a.txt 64
b.txt 32
soft 0 d
===sort by modified time===
soft d
b.txt
a.txt
/app/soft
===sort by size===
B.txt 1024
C.txt 1024
D.txt 1024
A.txt 64
===sort by size===
a.txt 64
b.txt 32
soft 24 d
/app
===sort by size===
t.txt 32
b.txt 32
app 17 d
===sort by size===
a.txt 64
b.txt 32
===sort by size===
t.txt 32
b.txt 32
app 12 d
```

What did I learn from this task

I learn some basic knowledge about file system design, like FCB entry and bitmap. Learn how to dynamically allocate the file. In the bonus task, I learn how to design a directory file system by using PARENT to link files.