Report for CSC3150 A4

Name：桂驰
ID：120090194

Environment

OS

NAME="CentOS Linux"

VERSION="7 (Core)"

ID="centos"

ID_LIKE="rhel fedora"

VERSION_ID="7"

PRETTY_NAME="CentOS Linux 7 (Core)"

ANSI_COLOR="0;31"

CPE_NAME="cpe:/o:centos:centos:7"

HOME_URL="https://www.centos.org/"

BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"

CENTOS_MANTISBT_PROJECT_VERSION="7"

REDHAT_SUPPORT_PRODUCT="centos"

REDHAT_SUPPORT_PRODUCT_VERSION="7"

VS

版本: 1.73.0 (user setup)

提交: 8fa188b2b301d36553cbc9ce1b0a146ccb93351f

日期: 2022-11-01T15:34:06.111Z

Electron: 19.0.17

Chromium: 102.0.5005.167

Node.js: 16.14.2

V8: 10.2.154.15-electron.0

OS: Windows_NT x64 10.0.22000

沙盒化: No

CUDA

nvcc: NVIDIA (R) Cuda compiler driver

Copyright (c) 2005-2022 NVIDIA Corporation

Built on Wed_Jun__8_16:49:14_PDT_2022

Cuda compilation tools, release 11.7, V11.7.99

Build cuda_11.7.r11.7/compiler.31442593_0

GPU

```
03:00.0 VGA compatible controller: ASPEED Technology, Inc. ASPEED Graphics Family (rev 41) (prog-if 00 [VGA controller])
        Subsystem: ASPEED Technology, Inc. ASPEED Graphics Family
        Flags: bus master, medium devsel, latency 0, IRQ 17, NUMA node 0
        Memory at 98000000 (32-bit, non-prefetchable) [size=64M]
        Memory at 9c000000 (32-bit, non-prefetchable) [size=128K]
        I/O ports at 2000 [size=128]
        Expansion ROM at <unassigned> [disabled]
        Capabilities: <access denied>
        Kernel driver in use: ast
        Kernel modules: ast
af:00.0 VGA compatible controller: NVIDIA Corporation Device 1eb1 (rev a1) (prog-if 00 [VGA controller])
        Subsystem: NVIDIA Corporation Device 12a0
        Flags: bus master, fast devsel, latency 0, IRQ 86, NUMA node 1
        Memory at ed000000 (32-bit, non-prefetchable) [size=16M]
        Memory at 3effe0000000 (64-bit, prefetchable) [size=256M]
        Memory at 3efff0000000 (64-bit, prefetchable) [size=32M]
        I/O ports at e000 [size=128]
        [virtual] Expansion ROM at ee000000 [disabled] [size=512K]
        Capabilities: <access denied>
        Kernel driver in use: nvidia
        Kernel modules: nouveau, nvidia_drm, nvidia
```

Execution Steps

Main Task

- cd Assignment_4_120090194/source
- sbatch slurm.sh

Bonus (Version 1)

- cd Assignment_4_120090194/bonus
- sbatch slurm.sh

Design Thought

Main Task

This task is aimed to simulate a file system in CUDA. We take the global memory as a volume. I divide it into three part: VCB(from 0 to 1*4KB), FCB Table(from 4KB to 4KB+1024*32B) and Storage(from 36KB to 36KB+4K*8*32B). The whole structure is showed as follows.

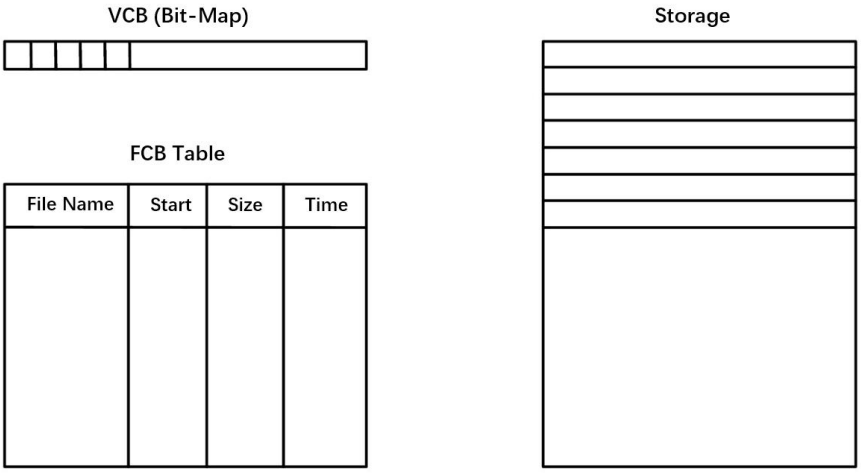

Figure 1: The whole structure for main task.

VCB: It is used to record the storage block information. It just a Bit-Map since the size of VCB is equals to the number of blocks in the storage. If block is empty, the corresponding VCB is 0, otherwise, is 1.

FCB Table: It is designed to storage the file information. I design a struct called **FCB entry**, consisting of File Name(char [20]), Start(Int), Size(Int), Time(Int). File Name is the file name. Start is the file storage start location. Size is the file size. Time is the modified time. The size of **FCB entry** is 32B. The **FCB Table** stores 1024 entries. And the order of the entry represents the create time.

Storage: It is designed to store file content. The program will store the content in terms of blocks. The block size is 32B.

Function Specific

fs_open:

1. Read: Search in the FCB table utilized file name. If find, return the file start position; If not, ERROR.
2. Write: Search in the FCB table utilized file name. If find, return the file start position; If not, create a new **FCB entry** in the table. And return the file start position.

fs_read:

From the file start position, load the data into output buffer.

fs_write:

1. If file is new created: update the **VCB, FCB entry,** allocate proper number of blocks and write the data into the **Storage.**
2. If file is old: compare the old block number and new block number and do the **data move** in the **Storage.** If the old block number larger than new block number, the data below target file should move up; otherwise, move down. And then, update the **VCB** and the **FCB Table**. The file below target file should update the start position. After, rewrite the content and clean up the remaining part of the last block. At last, update the size and time of the target **FCB entry.**

fs_gsys(RM):

Delete the file and release the file space.

1. Find the target file in the **FCB Table.**

2. Do the data move up in the **Storage.**
3. **FCB Table** update and move up.
4. **Target FCB entry** update.
5. **VCB** update.

fs_gsys(LS_D):

List all files name in the directory and order by modified time of files.

Iterate the whole **FCB Table** and order the file name according to the time.

Search the largest time every time and print.

fs_gsys(LS_S):

List all files name with size in the directory and order by size.

Iterate the whole **FCB Table** and order the file name according to the size.

Search the largest size every time and print.

Bonus

The bonus part is aimed to implement tree-structured directories. The whole structure is showed in figure 2.

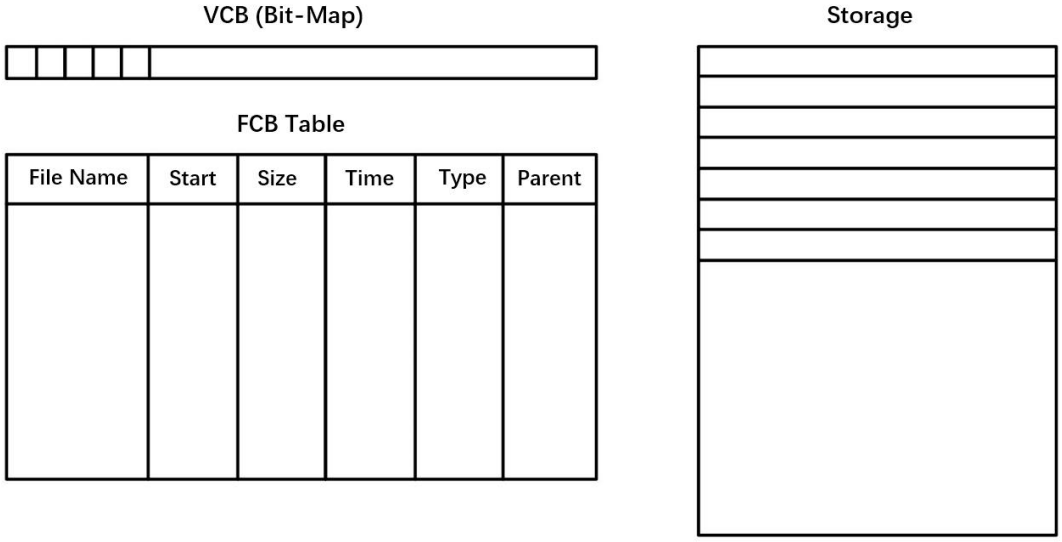

Figure 2: the whole structure for Bonus

The whole structure is similar to the main task except for the **FCB Table.** Now, the **FCB entry** is consists of 6 parts: File Name(char [20]), Start(Int), Size(Short), Time(Short), Type(Short) and Parent(Short). The type meanings the file type (dir: 1; file:0). The parent means the **FCB index** of the dir that the file belongs to. The relationship between directory and file is showed in the **Figure 3.** The directory will also store in the **FCB Table.**

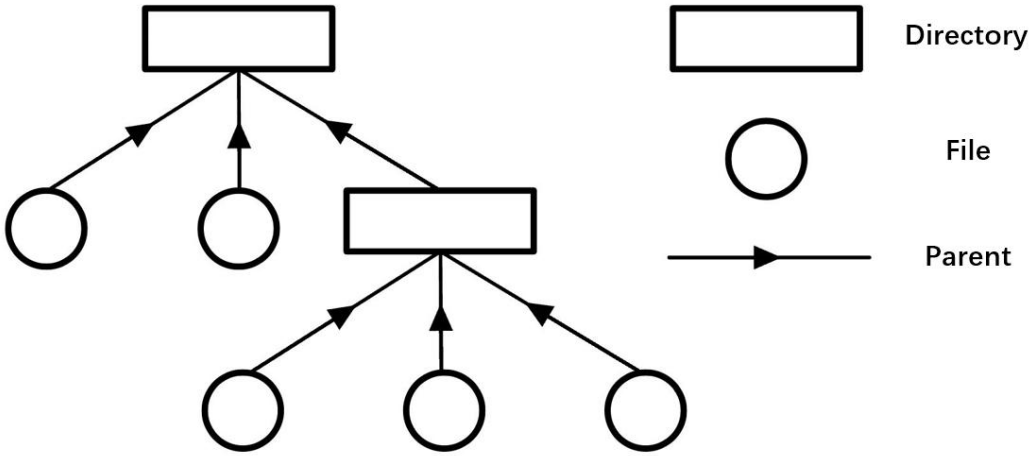

Figure 3: The relationship between Directory and File

Function Specific

fs_open:

Now the program only can open the file whose parent is the current dir. And the current dir should update its size.

fs_write and fs_read are the same as main task.

fs_gsys(RM):

When remove a file, the directory below the file will change the **FCB entry index**, thus, the parent of **FCB entry** which belongs to the directory should update. And the file's parent should update the dir size.

fs_gsys(MKDIR):

1. Update the **FCB entry** size of current dir.
2. Create a new **FCB entry.**

fs_gsys(CD):

Change the current dir if the CD target is exist under the current dir.

fs_gsys(CD_P):

Change the current dir to the parent of current dir.

fs_gsys(RM_RF):

Remove all the dir and file and subdir under the dir.

1. Find all the file including the file belongs to all subdir and subdir belonging to the dir recursively.
2. Remove file one by one.
3. Remove subdir one by one.
4. Remove dir and update the parent's size.

fs_gsys(PWD):

Find the absolute path.

1. Find its all the parent directory name.
2. Cat all the parent directory name with the file name and print.

fs_gsys(LS_D) and fs_gsys(LS_S) are similar to the main task.

Problem and Solution

Problem: How to design the File system?
Solution: The file system consists of three parts, VCB, FCB Table and Storage. I design a struct called **FCB entry** consists of many elements used to store the file information and interact with VCB and Storage. VCB is a bit-map, which is used to store the block available or not of the Storage. The Storage is used to store the data.

Problem: How to update the Storage when there is a file will be removed or rewrite existed file?
Solution: The program will do the **data move** in the Storage and **FCB table update (and move)** in the **FCB Table.** The specific strategy is list in the Design Thought.

Problem: How to implement tree-structed directories?
Solution: Add two new element in the **FCB entry**: type and parent. The type meanings the file type (dir: 1; file:0). The parent means the **FCB index** of the dir that the file belongs to. The relationship between directory and file is showed in the **Figure 3.** The directory will also store in the **FCB Table.**

Learning outcomes

1. **A better understanding of file system.**
2. **Learn how to design inverted VCB, FCB Table and Storage.**
3. **Learn how to use ssh in the vscode.**
4. **Learn how to use cuda and write simple cuda c program.**
5. **Self-learning ability improvement.**
6. **C program ability improved.**
7. **Learn how to implement the strlen, strcpy and strcat functions.**

Screenshot of program output

Main Task

Test Case 1

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

Test Case 2

```
===sort by modifed 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 modifed 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 modifed time===
*ABCDEFGHIJKLMNOPQR
)ABCDEFGHIJKLMNOPQR
(ABCDEFGHIJKLMNOPQR
'ABCDEFGHIJKLMNOPQR
&ABCDEFGHIJKLMNOPQR
b.txt
```

Test Case 3

```
===sort by modifed 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 modifed 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
```

```
===sort by modifed 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 modifed 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 modifed time===
*ABCDEFGHIJKLMNOPQR
)ABCDEFGHIJKLMNOPQR
(ABCDEFGHIJKLMNOPQR
'ABCDEFGHIJKLMNOPQR
&ABCDEFGHIJKLMNOPQR
b.txt
===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
......
```

```
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
```

```
===sort by file size===
EA 1024
~ABCDEFGHIJKLM 1024
aa 1024
bb 1024
cc 1024
dd 1024
ee 1024
ff 1024
gg 1024
hh 1024
ii 1024
jj 1024
kk 1024
ll 1024
mm 1024
nn 1024
oo 1024
pp 1024
qq 1024
}ABCDEFGHIJKLM 1023
|ABCDEFGHIJKLM 1022
{ABCDEFGHIJKLM 1021
zABCDEFGHIJKLM 1020

……
```

```
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
```

Test Case 4

```
triggering gc
===sort by modifed 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-0016
1024-block-0015
1024-block-0014
1024-block-0013
1024-block-0012
1024-block-0011
1024-block-0010
1024-block-0009
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
```

Bonus

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