CSC3150 Project 4 Report

1. How did you design your program?

Data structures I design this program using struct to represent dist structures:

```
struct FCB { // 32 bytes in total
    char name[20]; // 20 bytes to store name of the file
    int address = -1; // 4 byte to store block address of the file
    int size; // 4 byte to store size of the file (blocks occcupied)
    int time; // 4 byte to store time of the file
};

struct Block { // each Block takes 32 bytes
    char content[32] = { '\0' };
};

// this one is for bonus, but compatible for original problem
struct FS_node {
    FS_node *parent_node = (FS_node *)malloc(1000);
    FS_node *child_node = (FS_node *)malloc(1000);
    int childs = 0;
    struct FileSystem *this_FS = (FileSystem *)malloc(1085000);
    char *path = (char *)malloc(50 * sizeof(char));

    int size = 0;
    int time = 0;
};

struct FileSystem { // 1085440=1085476 byte
  // storage part: 32+1024+4=1060KB (1085440 byte)
  // 1024 FCBs, 32KB in total
    FCB *fcbs = (FCB *)malloc(1024 * 32);
  // 32768 Blocks, each Block takes 32 bytes, 1024KB in total
    Block *blocks = (Block *)malloc(100 * 32);
  // (converted to bitmap) 32768 bits indicating whether a block is occupied, 4KB in total
    unsigned int SuperBlock[1024];

  // pointer to the node containing this FS, only used in bonus part
    FS_node *node_ptr = NULL;

    // 4*9=36byte
    int SUPERBLOCK_SIZE;
    int FCB_SIZE;
    int FCB_ENTRIES;
```

```
    int STORAGE_SIZE;
    int STORAGE_BLOCK_SIZE;
    int MAX_FILENAME_SIZE;
    int MAX_FILE_NUM;
    int MAX_FILE_SIZE;
    int FILE_BASE_ADDRESS;
    int gtime;
};
```

Since the memory size when using struct is exactly the same as using volume, I think this is also OK. In addition, using struct improves the program's ablility to abstract the file system, easier to understand.

Note: FS_node is the node used in bonus, where a FileSystem represets files in a folder corresponding to a continuous memory area on the disk, and FS_node is a tree data structure representing directory. There is a pointer in FS to its node, and a pointer in node to FS. In this way we can find node / FS easily. Each time make a new folder, program request a area of disk (dynamic allocation memory in the program) for the folder. The benefit of doing so is that you don't need to search all fcbs to find files pointing to a folder, instead, only need to search the node's FCB (less FCBs need to check). This can save a lot of time.

Operations

- Open
  1. Find the corresponding FCB vie file name
  2. If found, return the FBC's index
  3. If not, find a empty FCB to store this newly opened file's info, and update bitmap
- Read
  1. Simply read each blocks' info
  2. Update FBC's modified time
- Write
  1. Simply write to each block
  2. Update FBC's size and modified time
- LS (by Date / Size)
  1. Declare a new array for sorted FBCs
  2. Copy all available FBCs to this array
  3. Use bubble sort to sort this new array (by Date / Size)
  4. Print info of the sorted array Note: for bonus, I don't count '\0' as a character, so folder size may differs
- RM
  1. Re-initialize its corresponding FCB to original state (this denotes that this FCB is not used)
  2. Re-initialize its corresponding bitmap to zero.
- RM-RF
  1. Generate the full absolute path

    2. Simply remove the folder's corresponding FS_node, this indicates the whole folder and its content are all removed
- MKDIR
    1. Generate the full absolute path
    2. Dynamicaly allocate a new FS_node representing the parent-child relationship
    3. Dynamicaly allocate a new FileSystem for the new node
    4. Attach this node to current node (folder), as well as attach the FS to the node
- CD
    1. Generate the full absolute path
    2. Locate the target child node (folder)
    3. set the current FS to its corresponding child. Since there is a pointer in FS pointing to the node, it is easy to move along the tree structure
- CD_P
    1. set the current FS to its parent.
- PWD
    1. Print out the path of this node, nothing needs to be explained

2. What problems you met in this assignment and what is your solution?

No many problems, except for to debug on CUDA and VS2017, it is fine for me. Although CUDA is annoying in terms of allocate memory (yes, when I try to allocate too much memory, it returns a null pointer to me).

3. The steps to execute your program.

Same as instructions, Ctrl + F7 to compile, and Ctrl + F5 to run.

4. Screenshot of your program output.

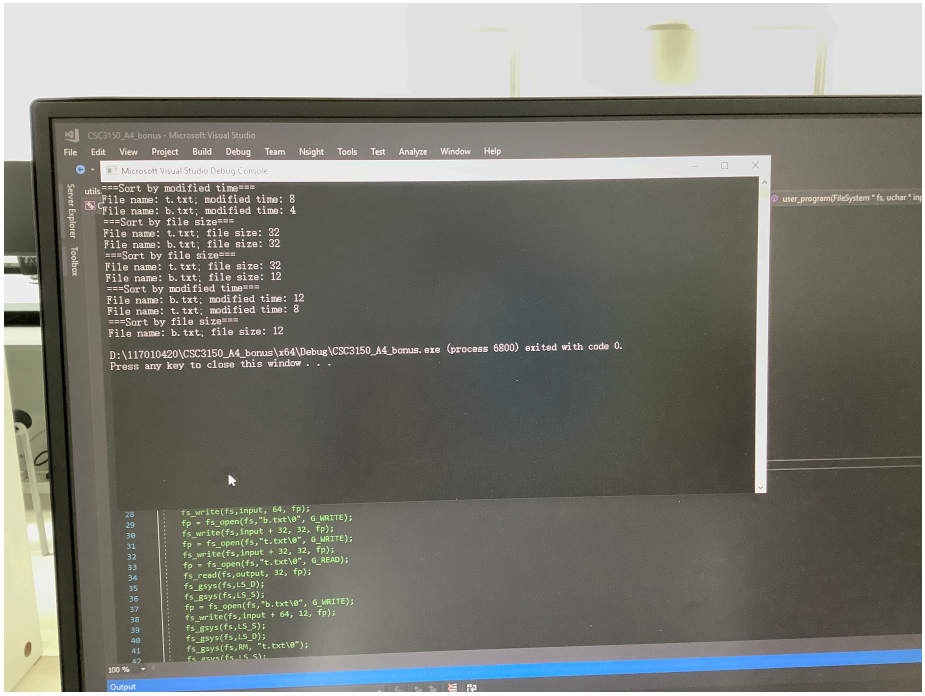

Figure 1: Test 1

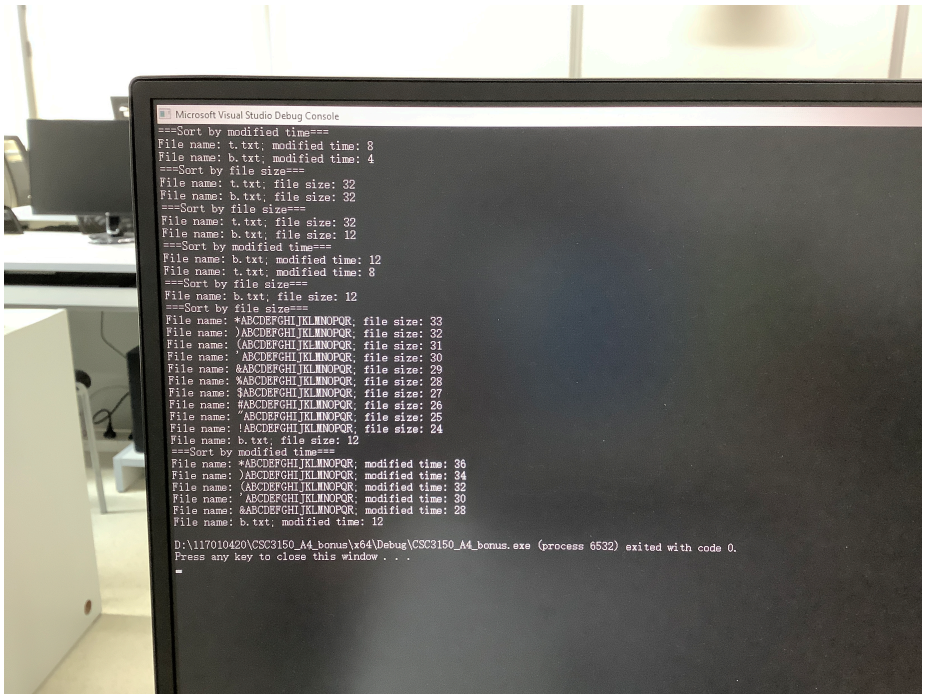

Figure 2: Test 2

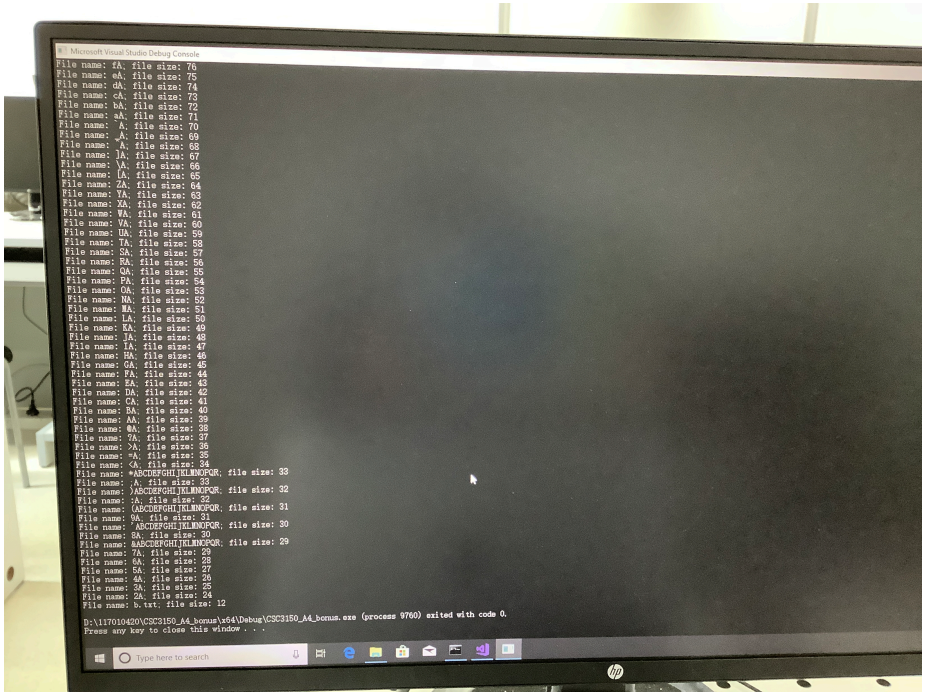

Figure 3: Test 3

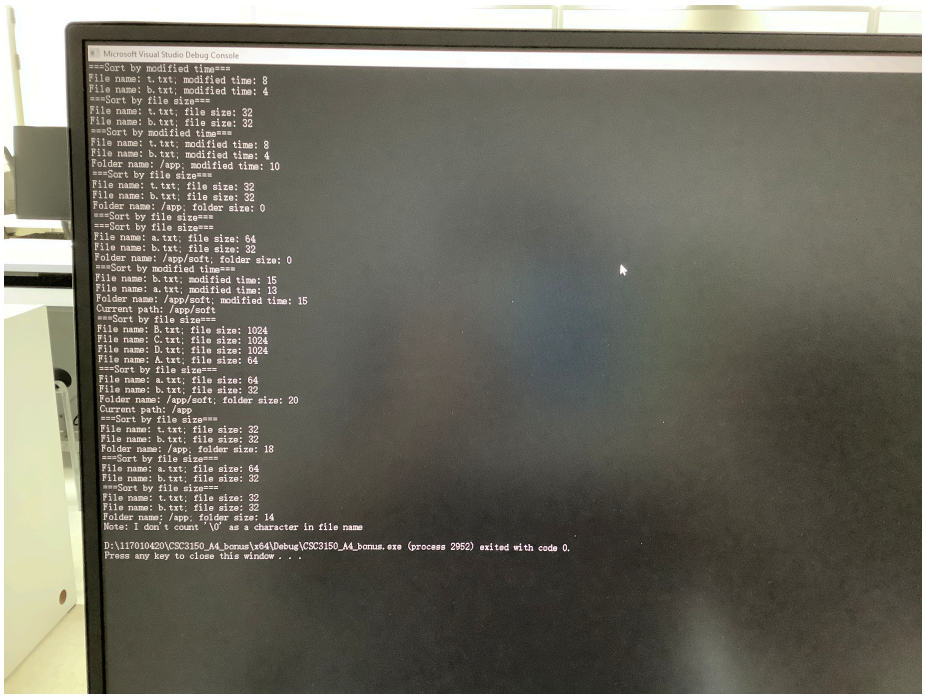

Figure 4: Bonus

5. What did you learn from this assignment?

I learned that CUDA is very difficult to use, so is a Windows system computer. As I mentioned, when I try to allocte too much memory in CUDA, it just returns a null pointer. I also wrote a C++ version project on my Mac OS and it works just fine on it.