Filesystem Documentation

The purpose of this project is to create your own filesystem, accordingly to file allocation table (FAT). There will be directory involved, which organizes the filesystem by have a root, parent and child nodes. The given source and header file are the “disk”, which is creating and accessing the virtual disks. Disk contains all the files, such as the file and meta-data (data blocks), FAT, root directory and boot sector.

So for my functions, I changed my mind instead of doing a tree, I used an array since I feel that it’s easier to implement and use memcpy() function to copy data over.

First, I test out is disk.c methods work, which they do. Then I created another C source file, “filesystem.c” along with the header file. In the filesystem.c, I included all the methods that was given to us. For the header file, filesystem.h, I included all the data information (meta data (for directory entry), FAT entry, boot sector (for directory entry). I did that as a typedef struct so this main information can have the detailed information. For example,

typedef struct FAT\_s{  
 unsigned int used; //Marks with FAT block are used  
 unsigned int end; //Tells where the last FAT pointer block is  
 unsigned int next; //Tells where the next pointer is  
}FAT;

the code above has unsigned int used, end and next as FAT entry’s detailed information. I have included detailed information for meta data and boot sector structs. I included the function implementation.

//Function implementation  
int make\_fs(char \*disk\_name, int size);  
int mount\_fs(char \*disk\_name);  
int unmount\_fs(char \*disk\_name);  
int fs\_open(char \*name);  
int fs\_close(int fildes);  
int fs\_create(char \*name);  
int fs\_delete(char \*name);  
int fs\_mkdir(char \*name);  
int fs\_read(int fildes, void \*buf, size\_t nbyte);  
int fs\_write(int fildes, void \*buf, size\_t nbyte);  
int fs\_get\_filesize(int fildes);  
int fs\_lseek(int fildes, off\_t offset);  
int fs\_truncate(int fildes, off\_t length);

I may have modified the make\_fs method by including size, where I want the user to enter in a specified size from the shell. In the filesystem.c, I included detailed information in the comments.

Since the methods have more information such as getting file name from directory entry or updating the directory entry, I will need another method for that. I included all the additional information in the filesystem2.c and its header file.

nt newFAT(FAT\*\* ft, BootSector\* bs); //creates new FAT entry  
int findDir(MetaData\* md, BootSector\* bs, char \*name); //finds dir entry by name  
void dirEntryDisk(MetaData\* md, BootSector\* bs, int i); //finds dir entry on disk  
void updateDir(MetaData\* md, BootSector\* bs, int i); //updates dir entry  
void getBlock(char \*buf, int i); //retrieves the block given by index and buf

In filesystem2.c, I have included more detailed information in the comments.

For the shell part, I included args and argv[] in the main and user can type in based on what the commands are available (in help section).

void makeFs(int args, char \*arg1, char \*arg2);  
void mountFs(int args, char \*arg1, char \*arg2);  
void unmountFs(int args, char \*arg1, char \*arg2);  
void openFs(int args, char \*arg1, char \*arg2);  
void closeFs(int args, char \*arg1, char \*arg2);  
void createFs(int args, char \*arg1, char \*arg2);  
void deleteFs(int args, char \*arg1, char \*arg2);  
/\*void mkdirFs(int args, char \*arg1, char \*arg2);  
void readFs(int args, char \*arg1, char \*arg2);  
void writeFs(int args, char \*arg1, char \*arg2);  
void getSize(int args, char \*arg1, char \*arg2);  
void lseekFs(int args, char \*arg1, char \*arg2);  
void truncateFs(int args, char \*arg1, char \*arg2);\*/  
void help();

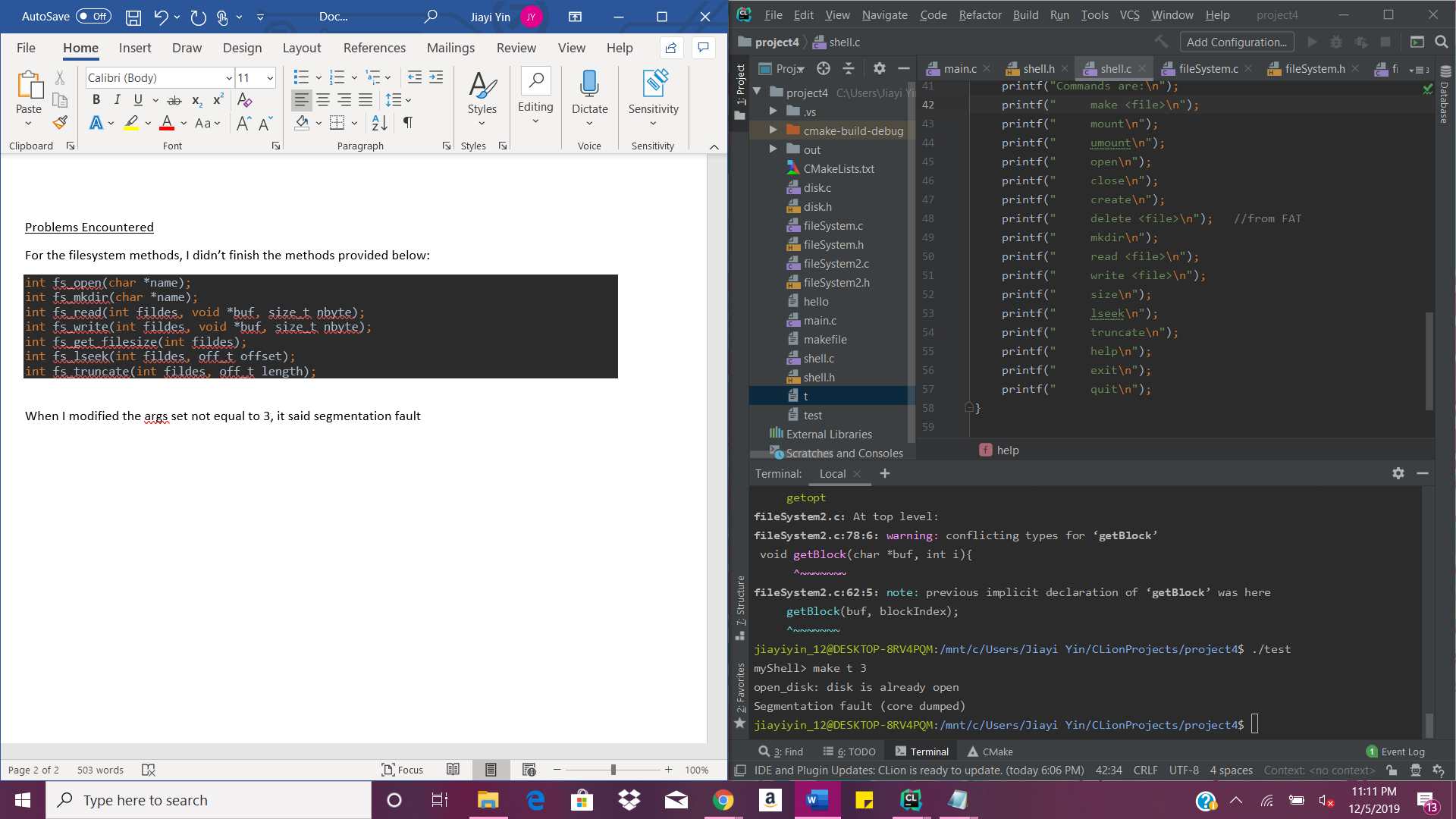
These are the methods that I have for shell.

Problems Encountered

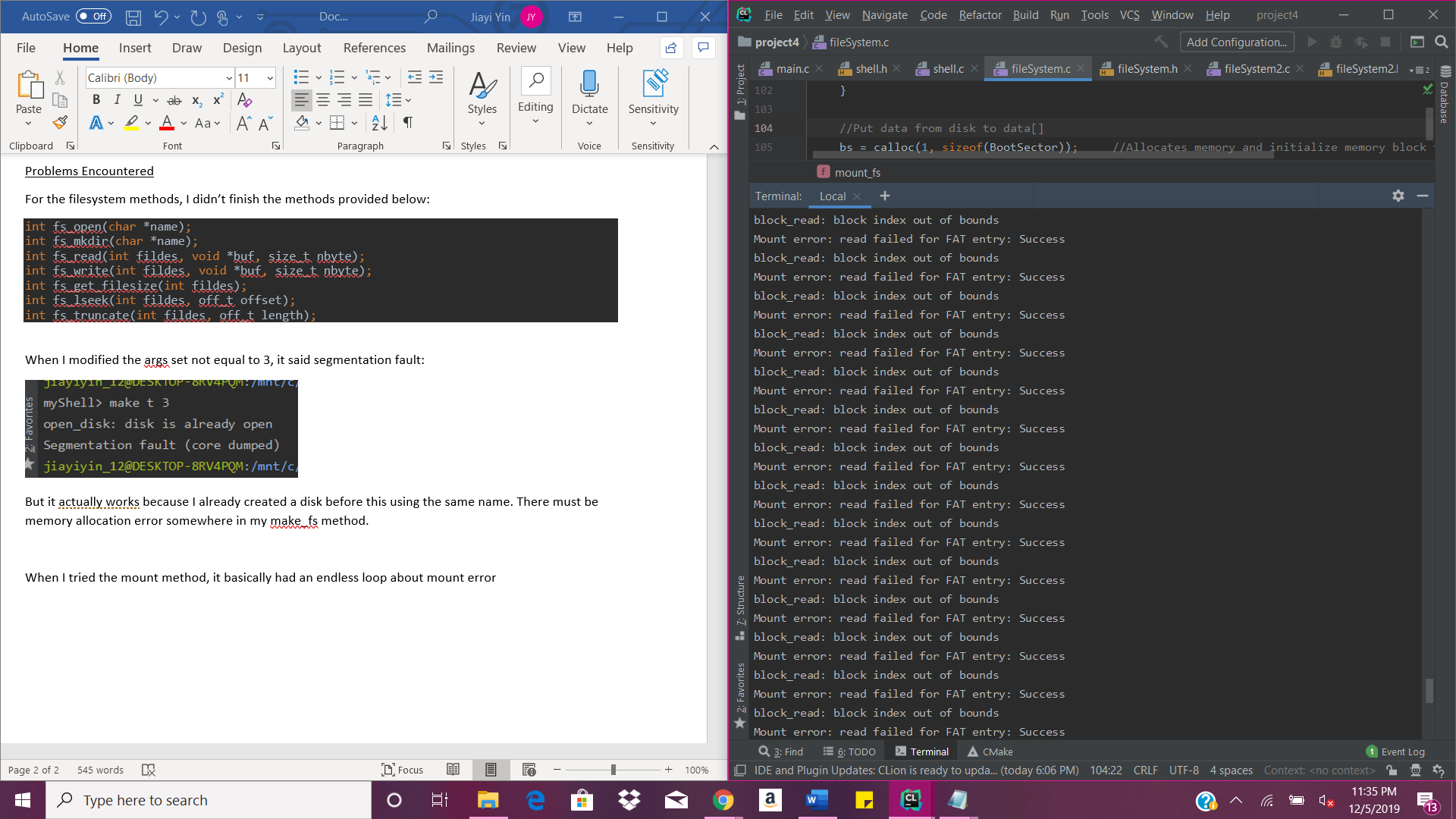
For the filesystem methods, I didn’t finish the methods provided below:

int fs\_open(char \*name);  
int fs\_mkdir(char \*name);  
int fs\_read(int fildes, void \*buf, size\_t nbyte);  
int fs\_write(int fildes, void \*buf, size\_t nbyte);  
int fs\_get\_filesize(int fildes);  
int fs\_lseek(int fildes, off\_t offset);  
int fs\_truncate(int fildes, off\_t length);

When I modified the args set not equal to 3, it said segmentation fault:



But it actually works because I already created a disk before this using the same name. There must be memory allocation error somewhere in my make\_fs method.

When I tried the mount method, it basically had an endless loop about mount error

I think it loops corresponding to block size. I’m not sure where the word “success” came from.