Project Name: **Virtual File Management System**

Author: **Chengjun Yuan** **cy3yb@virginia.edu**

Time: **Nov. 23 2015**

1 This project contains four source code files and one Makefile.

"disk.h"
"disk.c"
"p6cyuan.c"
"myApp.c"
"Makefile"

How to run it: first use command 'make' to compile it. Then use command './p6cyuan' to run it. The output should be like this:

```
file size of file fc = 78
The goal of this pro
ramming be
I love computer programming be
*** file fb has been deleted ***
*** Error: can not find file fb ***
status of file fb = -1
The goal of this project
I love computer programming be
Hi, there
I am glad this is the last project for the semester
```

Tips: the messages that start with "***" and end with "***" are the notice messages from "p6cyuan.c". Other messages are from "myApp.c".

2 The Data structure of File System

There are total 64 blocks and each block has the capacity of 16 bytes in the file system. The last 32 blocks are data blocks and the first 11 blocks are metadata blocks that store the file system management information (file directory, FAT, OFT).

Metadata (11 blocks)

Main	Block 0	File system name
File Directory	Block 1 – 4	File descriptor number, file length, file name, the first block number of file storage.
FAT	Block 5 - 8	File allocation table
OFT	Block 9 - 10	File open table

There are total 8 file descriptors in file directory and each one holds 8 bytes. The first byte indicates the descriptor number (0 – 7). The 2nd and 3rd bytes indicate the length of file as unit of byte. The following 4 bytes are used to store the file name (up to 4 letters). The last byte show the first data block of the file storage.

There are total 32 entries in FAT which correspond to the 32 data blocks. Each entry holds 2 bytes. The first byte indicates whether its corresponding data block is used (= 1) or not (= 0). The second byte indicates the pointer to the next data block. It equals to zero which means that current data block is the last data block of the file storage.

There are 8 entries in OFT, which means that we can open maximum 8 files concurrently. Each entry holds 4 bytes. The first bit in the first byte indicate whether the file with the same number is opened (= 1) or not (= 0). The rest 7 bits in the first byte indicate the data block where the current file pointer is located. The second byte indicate the offset position in the data block where the current file pointer is located. The rest two bytes indicate the overall byte offset of file pointer address based on the head of file.

3 Implementation of File System

int make_fs(char *disk_name);

Steps	Descriptions
1	Make disk for file system.
2	Open disk.
3	Write file system name into disk.
4	Close disk.

int mount_fs(char *disk_name);

Steps	Descriptions
1	Open disk
2	Read the metadata from the first 11 blocks of the disk. Save them in array meta.

int dismount_fs(char *disk_name);

Steps	Descriptions
1	Close all opened files.
2	Write metadata to the first 11 blocks of disk.
3	Close disk.

int fs_open(char *name);

Steps	Descriptions

1	Search the file descriptor.
2	Check whether it has already been opened or not.
3	Update OFT to mark this file opened.

int fs_close(int fildes);

Steps	Descriptions
1	Check whether fildes exist in directory.
2	Check whether fildes is opened.
3	Clear OFT to mark this file closed.

int fs_create(char *name);

Steps	Descriptions
1	Check the length of file name is up to 4.
2	Check whether the file exists or not.
3	Search the available directory entry.
4	Check whether there are available data block in file disk.
5	Save file name and block information into one entry of directory.
6	Mark the block is used in FAT

int fs_delete(char *name);

Steps	Descriptions
1	Check whether the file exists or not.
2	Check whether the file is opened or not now.
3	Find the first block in directory.
4	Free the listed data block of the file according to FAT.
5	Free directory entry.

fs_read(int fildes, void *buf_, size_t nbyte);

Steps	Descriptions
1	Check the validation of fildes..
2	Check whether the file exists or not.
3	Check whether the file is opened or not.
4	Obtain the current file pointer from OFT.
5	Read nbyte data and save them to buf_.

int fs_write(int fildes, void *buf_, size_t nbyte);

Steps	Descriptions
1	Check the validation of fildes.
2	Check whether the file exists or not.

3	Check whether the file is opened or not.
4	Obtain the current file pointer from OFT and get file length from directory.
5	Write nbyte data to data blocks.

int fs_get_filesize(int fildes);

Steps	Descriptions
1	Check the validation of fildes.
2	Check whether the file exists or not.
3	Get file length from directory.

int fs_lseek(int fildes, off_t offset);

Steps	Descriptions
1	Check the validation of fildes.
2	Check whether the file exists or not.
3	Check whether the file is opened or not.
4	Obtain the current file pointer from OFT and get file length from directory.
5	Check the boundary of file pointers.
6	Calculate the new position of file pointer.
7	Find the new position as the style of block and offset in block.
8	Update OFT with new file pointer.

int fs_truncate(int fildes, off_t length);

Steps	Descriptions
1	Check the validation of fildes.
2	Check whether the file exists or not.
3	Check whether the file is opened or not.
4	Check the validation of truncate length.
5	Get file length from directory.
6	Check the boundary of truncate.
7	Calculate the new file end of the truncated file.
8	Free blocks of the truncated part.