

project 4

1/4

(1) virtual disk

(1.1) 64 blocks

(1.2) 16 bytes/block

(1.3) range of block numbers: 0..63

(1.4) disk capacity: $64 \times 16 = 1024$ bytes

(1.5) disk name: 4 lowercase or uppercase letters

(2) directory

(2.1) single root directory (up to 8 entries, see below)

(3) files

(3.1) up to 8 files

(3.2) up to 32 blocks for data files (32..63)

(3.3) other 32 blocks are for metadata (0..31)

(3.4) maximum file size = $32 \times 16 = 512$ bytes

(3.5) file name: 4 lowercase or uppercase letters

(3.6) Open File Table (OFT): up to 4 opened files simultaneously (ie. size of OFT = 4)

(4) functions you need to design and implement

0..3

a. make-fs()

k. fs-lseek()

b. mount-fs()

l. fs-truncate()

c. dismount-fs()

d. fs-create()

e. fs-open()

f. fs-close()

g. fs-delete()

h. fs-read()

i. fs-write()

j. fs-get-file-size()

(5) functions provided to you

- a. `make-disk()`
- b. `open-disk()`
- c. `close-disk()`
- d. `block-read()`
- e. `block-write()`

(6) how to design "make-fs()" function?

- a. use "make-disk()" to initialize a new disk (i.e. store 0 in each byte on the virtual disk)
- b. use "open-disk()" to make the virtual disk available
- c. initialize superblock, directory, and FAT on disk (see below for disk layout)
- d. use "close-disk" to close the disk (i.e. make the virtual disk unavailable)

(7) how to design "mount-fs()" function?

- a. use "open-disk" to make the virtual disk available
- b. load directory and FAT into memory (use "block-read" to do it)
- c. create an OFT in memory

(8) disk layout (see next page)

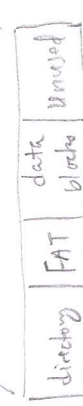
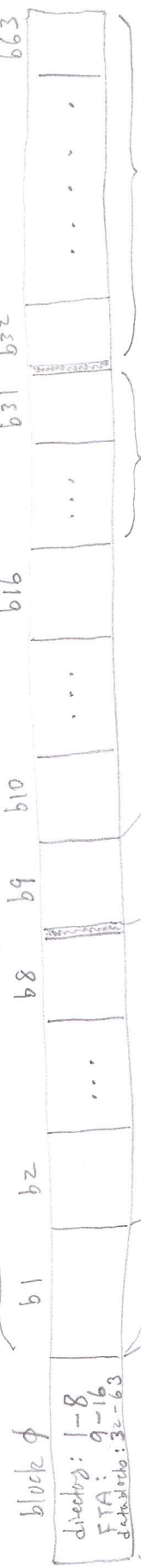
data blocks

FAT

directory

unused

512 bytes (characters)



4 bytes 4 bytes 4 bytes 4 bytes

01 08 09 16 32 63

char type

data

1st block number file



2 bytes 4 bytes 3 bytes (char) (char) (char) (unused)

1 byte (char) status

(free or allocated)

b2..b7

16 bytes each 16 bytes

b8

16 bytes each 16 bytes

1 byte 3 bytes status (char) (char)

(free or allocated) 4 bytes

b9

b10..b16

4 bytes each

block 32 block 33 block 34 block 35

to make 4 entries in 1 block

i.e. block numbers 32, 33, 34, 35 in block 9

block number 36, 37, 38, 39 in block 10

:

block numbers 60, 61, 62, 63 in block 16

a total of 32 data

block numbers

DISK Layout

(9) implementation (for "mount fs()")

(9.1) OFT (0..3)

status	file offset	index in the directory
0	0	file ptr
1	0	
2	0	
3	0	

an array of struct

(9.2) FAT (loaded from the virtual disk into memory)

status	block number
32	0
33	0
34	0
35	0
36	0
:	:
63	0

FAT
(an array of struct)

Each entry has 4 bytes

e.g. if blocks 32, 43, and 35 were allocated to a file, then if file ptr/offset = 23, then it is pointing to $23 - 16 = 7$ in block 43

(9.3) directory (loaded from the virtual disk into memory)

file name	file length	1st block number
0	0	
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	

directory
(an array of struct)

each entry has 16 bytes (i.e. 1 block)