Computer Science

COSC 439

Group 5

File System Management

•Project Organization

Team member's information

• Roles in Team Project

•Objective in Team project

By doing this project, we will become familiar with the file system management operation and how it works actually. File system is the only part in the operating systems that has the logical structure of it visible to users and the physical structure hidden from the users. This system communicates with the file management section for accessing, storing and releasing, deleting.

•Description in Team project

File system management is based on OS. It consisted of files and directories that are defined by owner. User can modify and create and delete and sort files, directories on storage if user has authority. So File system affects system in

many ways. Therefore the goal of File system is to enable you to manage files quickly.

•**Approach in Team project**

- We will make a file management program based operating system.
- We will create
  - ✓ The ability to show the lists.
  - ✓ The ability to make and delete a file and a directory.

Contents

1.Object

2.Detail contents

1. Object.

A simple file system is implemented, and we understand functions, roles of file system.

2. Detail contents

2.1 Implement of file-system simulator “filesystem_sim”

2.1.1 layout of file-system

- The goal of this project is design, implement of file system of simple Unix/Linux. The file system assumes the following detail, we have easy access to file-system.

- The file system exists on the disk to the size of 128KB(file-system)+32KB(for log).
- The file system has only one root directory. Sub-directory does not exist.
- File system supports 16 files of maximum.
- The maximum file size is 8 blocks. Each block is 1KB.
- Each file has a unique name. File names are allowed Until the 8 character.
- 32KB is size for log.

- The layout of the disc of 128KB is as follows.

- First 1KB block called a super block. This saves free-block list and inode(index node) of each file.
- At rest 127KB, each 1KB block saves data-block of files about file-system. (127 data blocks of 1KB)
- The exact structure of a super block is as follows.

- ✓ The first 128 bytes save lists of block list. Each entry in the list, the corresponding block display free or in use. (If i-th byte is 0, it’s free. If i-th byte is 1, it’s in use. At first, every block except super-block is free.)

- ✓ The part after the free block list is 16-inode. File in the file-system has one inode. Therefore file-system has 16 files. At first, inode is free. Each inode has the following information:

```
char name[8];//file name

int size;// file size

int blockPointers[8];// direct block pointer (defines that file has up to 8 blocks of 1KB)

char time[4]; // The last modification time of file

(ex : save at 1833 display on 18:33)

int used;// 0 -> inode 가  free, 1 -> in use
```

- ➢ Each inode has a size of 52 bytes(8+4+32+4+4). Because there has 16 inode, size of occupying by the inode is 832 bytes. Free/used block information is 128 bytes. Size of occupying by super-block is 960 bytes, So 1024 bytes (1KB) does not use all.
- ➢ Layout of 32KB disk for log is as follows:

```
char name[8];// file name

int logPointers;// the place is saved Block pointer actually

char data[1024];// the data is saved actually

char complete;// whether this is complete or not

chat commit; // whether this commit or not
```

- log can register up to 31. since 31th, log overwriting the old information.
- complete applies only to the log. Default value is ‘n’, and updates ‘c’.
- commit applies to the actual file-system. Default value is ‘n’, and updates ‘m’.

2.1.2 File system operation

- File system realizes as below

- create(char name[8], int size, char log): This function makes new file named name parameter and sized size parameter. Log parameter is option. (When new file is created, it is accurately made as specified.)
- delete(char name[8]): This function deletes file.
- read(char name[8], int blockNum, char buf[1024]): This function inserts buf you specified after reading block you specified. “blockNum” is possible from 0 to 7.
- ls(void): This function shows file name and file size in file system.
- infodiskfree(void) : This function shows free block list in file system in order.

- We should suppose that disk is for writing file of 128KB. To make this disk, you use below example.

Product example)
filesystem_sim -c disk0

- If you add –c and disk0 options, disk0 file of 128KB will be made in this directory. While disk0 file is made, also disk0 file is formatted. Format initializes all of blocks and sixteen inodes are marked as ‘free’.

- File system should be persistent. If you reboot computer, all files in file system should be accessible.

- filesystem_sim program can process commands in disk you make. Below commands are formats.

Input File format)	
M diskName	// Disk name operating emulation.
C fileName Size	// Create file
D fileName	// Delete file
L	// Show all file names and sizes in disk
I	// Show free/used block list in disk in order

- Example of ‘L’ command (L command equals is command)

Output display)
Name Sizetime file1 318:33 file2 718:32

lab.c 710:18 ...

- Example of ‘l’ command

Output display)
0100011100000000111010000000.....

- Example of log command

Output display)
Name logpointdata complete commit file1 62sda1234523...c m file2 12.....c m lab.c 24.....c m ...

- Example of input file

Input example)

```
# filesystem_sim -c disk0

# filesystem_sim -c disk1

# filesystem_sim

> M disk0

disk0> C file1 3

disk0> C lab.c 7 L

disk0> L

...

disk0> I

...

disk0> log

...

disk0> C file2 4

disk0> R file1 1

...
```

```
disk0> D lab.c

disk0> L

...

disk0> M disk1

disk1> L

..

disk1> C file2 4

disk1> R file1 1

...

disk1> D lab.c

....
```

- disk0 and disk1 are absolutely distinguished. In other words, file1 in disk0 doesn’t equal file1 in disk1. These are different files. ‘-c’ option makes you can create and control many disks.
- Whenever you start filesystem_sim, you have to execute “M diskname” command. This makes you use disk you’d like to handle.

2.1.2 File System Calculation

- In file system, the computer can make maximum 16 files and one of files can have maximum 8KB. Therefore, the size of 16 * 8KB is 128KB, but as file includes super block, it can be 129KB. Some people think it is hard to handle all of the files, but the fact that file system support maximum 8KB doesn't mean all of files can be 8KB. In fact, most files are much smaller than system allows. Whenever there is a request of "create", the computer should check there is enough space. If the space is not enough, the computer print error message "not enough space on disk". Also, if people try to read, write, and delete with nonexistent files, the message "file not found : filename" comes up on the screen. Disc is same to file.

- When the computer handles super block, it doesn't read and write in ASCII way.

- It is the true that most operation handles the only super blocks(free block list and inode), but the calling of read and write reads and writes the appointed data in the Disc. In writing, you should be careful about using dummy buffer 1KB and data used from you can be read from the Disc later. The content of dummy buffer can be used on each team.

2.2 Compaction implementation

- In prepared file system, the computer makes compaction which deletes existent fragmentation. Compaction is utility shape that user can call. Whenever free space fragmented, file blocks are dotted at random on the Disc. The prepared compaction makes the files on the Disc to situate continuously. All of free space sends to end of the Disc. Any algorithms can be used to eliminate any fragment. The constraint is that you can't use the memory buffer with over 20% in total amount of Disc.

Implement example)
#compaction disk0

- The error message will be printed when the input formal is wrong, or the file is nonexistent.

2.3 Dynamic file system implementation

2.3.1 Lay out of file system

- Implement that when the computer create the first Disc, it can determine amount in dynamic way.
- As the amount is dynamic, the file system supposes that

- There is the only one root directory. The under directory is not existent.
- Each file has distinct name.
- The size of each block is 1KB.
- The size of log is a quarter of file system size.

- As lay out from 3.1.1, the amount of Disc lay out decides 1KB, but it can use 129KB, so super block, the size of inode, the number of block in each files, the size of maximum file, the number of usable block can change at each team. Therefore, it can be marked after the Disc is created. Also, you should write in the report why you choose like that. The amount of file system is within 16KB.

2.3.2 File System Calculation

- The all calculation of the first sub program can be used.
- The Disc is created in dynamic amount.

Example) Use the method to create 16M and 128K Disc.
Dynamic_filesystem_sim –c disk0 16M

Disc amount : 16MB (Output can be changed as each team)

…

Dynamic_filesystem_sim –c disk1 128K

Disc amount : 128K ((Output can be changed as each team)