Tuan Nguyen

File System

Introduction

- A File system used to store and organizes data. Within the system, we can access individual file by called their name and the place they are stored. For example, home/dir1/file1. The file can contain information such as the size of the file, as well as its attributes, location and hierarchy in the directory metadata. Metadata can store and retrieve file such as, date created, data modified, data of access, user ID of file creator, and access permission.

Design

-Size of virtual disk: the user can choose to use 2MB , 3MB, 5MB, 10MB to store file

-I will have first 10 blocks for file allocation table (FAT 32), each block size 512, and total of 5120 bytes. Each entry will store pointer to next block (2 bytes). The directory entry table(root directory) will start at block 10. Block 10 and 11 will store Metadata: file name(13 bytes), file extension(3 bytes), creation time (2 bytes) , create date (2 bytes), modified time (2 bytes), modified data (2 bytes), last access (2 bytes).The total of 24 bytes for each new file created. From block 12 to 4096 will store data. Data region is 4084 block = 2091008 bytes. The total block is 4096 and total disk drive is 2MB

Calculation

1 kilobytes = 1024 bytes

½ kilobytes = 512 bytes

1MB = 1024 kilobytes = 1048576 bytes

2MB = 2048 kilobytes = 2097152 bytes

2097152 bytes / 512 bytes = 4096 block

- 4096 block X 1/2 kilobyte per block= 2048 kilobyte = 2MB = 2097152 bytes

file table:

name parent date pointer/ directory

/ (root directory) none 2/3/18 directory

hello.txt / 1/29/18 index 1

in.txt dir1 3/09/19 index 2

image hello.txt 2/3/19 index 3

/(root directory)

dir1 hello.txt

in.txt image

Implement

The file system can perform these following function

1) create:

-search for file in root directory, if already exist generate error

-else: create an entry, update Dir Table, update FAT, add the filename to the current

directory block

-if the file is a directory, update currentDir to indicate we are in the directory after

creation

2) delete:

-search for file in root directory, if not exist, generate error

-remove entry from dir Table

-Update FAT and the filename in current directory block

3) open:

-find the first block of the file

4) close: -go back to root directory

5) read: -from the 1st block, navigate all the other blocks and read it into a buffer

6) write: -from the 1st block, navigate to the last block, find new empty block to insert

data, then update FAT, meta data.

Pseudocode

//information of data entry in struct

struct metadata{

char filename[10]; //13 byte for file name

char file\_exten[3]; // 3 byte for file extention

unsigned int create\_time; bbb// 2 byte for creation time

unsigned int create\_date; // 2 byte for create date

unsigned int modified\_time; // 2 byte for modified time

unsigned int modified\_date; // 2 byte for mofifies date

unsigned int last\_access; // 2 byte for last modified time

//all used to store unsigned integer

};

//pointer to data entry in directory table

struct metadata \*dir\_table;

struct Fat{

unsigned int next; // next block 2 bytes

};

struct Fat \*fat; // pointer to fat table

// create function can create new file or new directory

void create(char \*filename){

struct metadata dirEntry; //struct for directory entry

struct Fat fat; // struct for fat

char \*f1 // store file name

char ch;

ch = ‘.’ // current directory

int entryIndex, //entry index in directory table

int blockFree // free block

if((filename= (f1){ file is exist

printf("file is already exist\n");

return;

}

else ( create a file entry )

//allocate space for file

//update that space in block

// if it a directory

//create a new directory entry with the following content:

-dirEntry.name = filename;

-dirEntry.create\_time = time(0);

- dirEntry.last\_access = time(0);

-dirEntry.modified\_time = time(0);

-dirEntry.start\_blocks = blockFree; //start entry into empty block

-dirEntry.size = 0;

use memcpy to update directory table

memcpy(dir\_table[blockFree], dirEntry, sizeof(struct Entry)

}

//delete a file

void delete(char \*filename){

struct metadata emptyEntry

int blockStart; // use to delete from

int blockEnd; // use to delete to

char deleteName; //store name that matches user input

char \*data ; // data that delete

if((deleteName != (filename)) ){

printf("File cannot be found\n");

return;

}

else( remove entry from directory structure table

//replace entry in directory table with empty entry

memcpy(&dirTable[dirIndex] , &emptyEntry , sizeof(struct Entry));

for(int i = blockStart ; i < blockEnd; i++){

data[i] = 0;

}

// update space in directory table

//free block entry

}

//read a file

void read(char \*filename){

int blockStart = 0; // initial block

int blockEnnd // last block

char readFile // name of file to read

int fatIndex = 0 // index in fat table

char buffer[512] // size of file

//index in FAT of the file

blockStart = (fatIndex – (start of data = block 12)) x 512 bytes

//calculate the total size reading

blockEnd = blockStart + 512 bytes;

// file does exist

if((realdfile = filename))

// start read a file

//read block in file one by one

for(int i = blockStart ; i < blockEnd ; i++ , j++){

memcpy(&buffer[j] , sizeof(char));//read bytes to the buffer of size 512

printf("%c" , buffer[j]); // print character of each data reading

//update block is reading

else{ // file doesn’t exist

printf("File cannot be found\n");

return;

}

// write a file

void write(char \*filename , char \*writeData){

int dataSize // size data written

int blockStart = 0; // initial block

int blockEnd // last block

// length of written data

dataSize = strlen(writeData);

if(( finename is not found){ //search for exist file to write,

printf("File cannot be found\n");

return

else{// start written

blockStart = (freeBlock - START\_OF\_DATA(block 12) \* 512; // start point

blockEnd = blockStart + 512; // last point

for(int i = blockStart ; i < blockEnd ; i++ , size++){ //continue to write until the end of the string

memcpy( &writeData[size] , sizeof(char)); // copy each block of data into written file

}

dataSize -= size; // decrement size

}

//update allocate space

//update meta data

open a file

int open(char \*filename){

int dirIndex; //index in directory table

int fatIndex; //index in fat

char f1; // store pointer to matches file name

if( dirIndex = searchFile(filename) == -1){

printf(“file can not be found)

else{

printf(“file found”);

fatIndex = dirtable[dirIndex].start\_block(block 0) // open file in first free block in Fat table

Search file use to find a file name in root directory

int searchFile(char \*filename){

char name[13];

for(int i = 0 ; i < 512 bytes ; i += 32bytes (fat32 entry of 32 bit){ // search each block in table

if(datamap(array to map) != empty and datamap(array) != ‘.’ (current directory

memcpy(&name[i], datamap [i +1], sizeof(char);

}

return 1;

close a file

void close(){

set current directory = 0; closes a file descriptor

, so that it no longer refers to any file and may be

reused

int currentDir;

currentDir = 0;

}

//find free data Block, return the block number

int findFreeBlock(){

for(int i = 0 ; i < DATA\_SIZE(2091008) ; i += 512){ // loop through data region

if(dataMap[i] == 0){ // datamap[i] = 0 mean free block

return START\_OF\_DATA(block 12) + i / 512;

}

}

return 0;

}

//find used data Block, return the block number

int findUseBlock(){

for(int i = 0 ; i < DATA\_SIZE(2091008) ; i += 512){ // loop through data region

if(dataMap[i] == 1){ // datamap[i] = 1 mean use block

return START\_OF\_DATA(block 12) + i / 512;

}

}

return 0;

}

testing

// the drive user will use is 2MB

//next Display a list of function options

Here are your options:

-Create

-Delete

-Read

-Write

-mkdir

-List

-Open

-Exit

-Close

//For example : user select create

//ask to Enter the name of file to create: hello.txt

//if create successfully: print success create file

//else print: can’t create file because of matching name or out of space

//For example : user select delete

//ask to Enter the name of file to delete: hello.txt

//if delete successfully: print success delete file

//else printf (can’t delete hello.txt)

//For example : user select read

//ask to Enter the name of file to read: hello.txt

//If user hadn’t written in hello.txt, then a black line appear on display

// else record every single character that user written in hello.txt

//For example : user select write

//ask to Enter the name of file to write: hello.txt

//if filename can’t found , print “can’t find hello.txt to write”

//else user can start write to hello.c

//For example : user select mkdir

//ask to Enter the name of directory to make: dir1

//if success, printf “successfully create directory dir1”

else printf(can’t create directory )

//For example : user select ls

//list the content of current directory

for example: hello.txt

//For example : user select open

//ask to Enter the name of file to open: hello.txt

//if success, printf “successfully open hello.txt”

else printf(can’t open hello.txt)

//For example : user select exit

//printf (file system is exit);

//exit the program

//For example : user select close

//ask to Enter the name of file to close: hello.txt

//if success, printf “successfully close hello.txt”)

else printf(can’t close hello.txt)

// Program will print the path that user current at

ex: home/dir/file.txt

//User can print data, time , year of specific file

ex: hello.txt 1/30/19