Part 2: File Systems

Team Member

shengbo lou: 28530995
zhongce ji: 28551884
yucheng lu: 28411887

Compile

Go to the lab03/part1 folder

- compile: `make`
- run: `./main`

Design

We use the C language to complete the second part of the lab3. For this part, we are requested to implement five main method which are create, read, write, delete and ls of the file system. So we start from creating the interface which includes five method and we implement them in part2.c. Then we read the input and run the program in the main.c.

- part2.h
  This is the interface of the part2, in the file we have 5 methods that we need to implements:

```
void f_create(char name[8],  int size);
int f_delete(char name[8]);
void f_read(char name[8], int blockNum, char buf[1024]);
void f_write(char name[8], int blockNum, char buf[1024]);
void ls();
```

Those are five method we need to implement for the file system. First the `f_create` will take three arguments, `name` and `size`. The `name` is the file name that we need to create and `size` is the file's size. Then `f_delete` will take one argument which is `name`. It will delete the file whose name is `name`. `f_read` takes three arguments, `name`, `blockNum` and `buf`. The `name` indicates the filename we need to read. `blockNum` indicates the specific block we need to read. `buf` is a buffer that store the data read from the block.`f_write` takes three arguments,`name`, `blockNum` and `buf`. The `name` indicates the filename we need to write. `blockNum`indicates the specific block we need to write. `buf` is a buffer that store the data we need to write to block. `ls` will list all the files and its size.

- part2.c
  In this file, we will implements the method we defined in the interface:

`f_create`: For this method, it takes two arguments. One is file name and the other one is the file size. First we read first block which is super block from the file system:

```
char *buf;
buf = (char *) calloc(1024,sizeof(char));
if(read(fd,buf,1024)<0){
  printf("error: read failed\n");
}
```

we use calloc to create a buffer whose size is 1024 bytes so we can use it to read
the first block. Then we check the free list to find any free block for this file. If
we find the free block we will get the number of that block and store it in the `block_pointer`.
Then we will check the inode to find a free inode to store the file name and size and its
`block_pointer`. we use `inode_used` helper method to get the index of the used, so that
we can check whether the inode is used or not. if the inode is free, then we go back to
the beginning of the inode by using the `name_index` to write the name in the inode.
After that we write the size into the inode. Then we write the `block_pointer` into the inode.

`f_delete`:For this method, it will take only one arguement which is name. First we will create
a buffer whose size is 1024 to read the super blcok from the file system. Then we check each
inode's
name. we use namex to store the name which is read from the inode, and the equation

```
int current_location = 128+(i*48);
```

will give use the first character of the name. if name is same as namex,Then we
have to get the block_pointer of that inode. So here we use

```
int tmp = current_location+15;
```

to jump to the first block number in the block_pointer. Then we use for loop to get
all block number. After that we clear this inode by setting all data in inode to 0, start
from the beginning of the inode. After that we go back to the free list to clear the free
list. After that we write the new buffer back to the file system.

`f_read`:For this method, it will take three agruments which are name,`blockNum` and
buf. Frist we will create a buffer whose size is 1024 to read the super block from the file
system. Then we check each inode's name. we use namex to store the name which is read from the
the inode and and the equation

```
int current_location = 128+(i*48);
```

will give use the first character of the name. if name is same as namex, then we get the
`block_pointer` of that inode
then we get the specific block we need to write by using :

```
int x = f[current_location+11+4*(blockNum+1)];
```

`current_location` is the beginning of the inode, then we jump 11 space to the block pointer,
then we jump another 4*(blockNum+1) spaces to the position that we want to get the block
number. Then we use `lseek` to that block and use read to read the data from that block
into the input buffer.

`f_write`:For this method, it will take three agruments which are name,`blockNum` and
buf. Frist we will create a buffer whose size is 1024 to read the super block from the file
system. Then we check each inode's name. we use namex to store the name which is read from the

the inode. if name is same as namex, then we get the block_pointer of that inode
then we get the specific block we need to write by using :

```
int x = f[current_location+11+4*(blockNum+1)];
```

current_location is the beginning of the inode, then we jump 11 space to the block pointer, then we jump another 4*(blockNum+1) spaces to the position that we want to get the block number. Then we use lseek to that block and use write to write the data into that block.

ls:For this method, it will list all files and its size. First we will create
a buffer whose size is 1024 to read the super block from the file system. Then we read the super block from the file system. Then we create name to store the name that we read from the super block. Then we use the helper method inode_used to get the used position. Then we check if the node is free, if the node is not free, we go back to the beginning of the inode to get the file name. And then get the size after the file name. Then we print out the output.

inode_used: This is a help method, which takes an input which means the ith node
This method will find the position of the used in that inode by using the equation:

```
127+48*input
```

The first 128 is the freelist and used is the last 4 bytes in the inode so we jump 48 bytes to the used position.

- main.c
  In the main, First we will read the name of the file system from the input file and use the system to create the file system. Then we read each action from the input. We will first read the type. if type is C which means create, then we have to read the name and size. Then call the function f_create to create the file in the file system. if type is W which means is write, then we have to read the name and blockNum. And then create a dummy buffer which contains some data need to be write into the block.
  Then call the function f_write to write the file in the file system.if type is R which means read, then we have to read the name and blockNum. And then we create an empty buffer
  to store the data that read from the block. Then call the function f_read to create the file in the file system. if type is D which means delete, then we have to read the name. Then call the function f_delete to delete the file. if type is L which means list, then we call the function ls to list all the file and its size.