同济大学计算机系

操作系统课程综合实验报告

学　　号 2153683
姓　　名 郭嘉
专　　业 计算机科学与技术
授课老师 方钰

装 订 线

目录

装 订 线

一，需求分析

1.1 实验目的

本次实验为构造一个类 UNIX 二级文件系统，并且能与用户进行交互，实现基本文件操作，从而掌握 UNIX 文件系统结构。

使用一个普通的大文件（如 c:\myDisk.img ，称之为一级文件） 来模拟 UNIX V6++的一张磁盘。磁盘中存储的信息以块为单位。每块 512 字节。并在该逻辑磁盘上定义二级文件系统结构。

superblock	inode区	文件数据区

（1）磁盘文件结构

- SuperBlock 结构
- Inode 结构
- 文件 File 结构
- 目录 Directory 结构

（2）高速缓存结构

- Cache 设计
- Cache 分配与回收
- Cache 读取与写入

（3）其他结构设计

- 索引结构

装

订

线

1.2 输入输出形式

（1）输入形式：通过类 linux 指令与用户进行交互。如 help，format，ls，mkdir，cd，create，open，write，read，seek，close，delete，exit 等指令。

```
*****************************************************************************************************
*                                                                                                   *
*                                        Unix文件系统模拟                                           *
*                                                                                                   *
* [usage]:                                                                                          *
* help                                  [功能]:命令提示                                             *
* fformat                               [功能]:格式化文件系统                                       *
* ls                                    [功能]:查看当前目录内容                                     *
* mkdir <dirname>                       [功能]:新建文件夹                                           *
* cd <dirname>                          [功能]:进入目录                                             *
* fcreate <filename>                    [功能]:新建文件filename                                     *
* fopen <filename>                      [功能]:打开文件filename                                     *
* fwrite <fdnum> <infile> <size>        [功能]:从文件infile写入fdnum文件size字节                    *
* fwrite <fdnum> std                    [功能]:从屏幕写入fdnum文件size字节                          *
* fread <fdnum> <outfile> <size>        [功能]:从fdnum文件读取size字节，输出到outfile              *
* fread <fdnum> std <size>              [功能]:从fdnum文件读取size字节，输出到屏幕                  *
* fseek <fdnum> <step> begin            [功能]:fdnum文件指针从开头偏移step                          *
* fseek <fdnum> <step> cur              [功能]:fdnum文件指针从现有位置偏移step                      *
* fseek <fdnum> <step> end              [功能]:fdnum文件指针从末尾偏移step                          *
* fclose <fdnum>                        [功能]:关闭文件句柄为fdnum的文件                            *
* fdelete <filename>                    [功能]:删除文件文件名为filename的文件或者文件夹            *
* exit                                  [功能]:退出文件系统                                         *
* fsave                                 [功能]:保存文件系统，cache写回（测试命令）                  *
* 注意：退出文件系统要使用exit命令！不能直接退出！                                                  *
*****************************************************************************************************
```

（2）输出形式：数据通过控制台输出，或者可以写入到文件中。提示信息通过控制台输出。

1.3 程序功能

（1）用户接口

- help：命令提示信息
- fformat：格式化文件系统
- ls：查看当前目录的结构信息
- mkdir <dirname>：在当前目录下新建文件夹 dirname
- cd <dirname>：进入目录 dirname
- fcreate <filename>：在当前目录下新建文件 filename
- fopen <filename>：打开当前目录下的文件 filename，返回文件句柄 fdnum
- fwrite <fdnum> <infile> <size>：从文件 infile 写入句柄为 fdnum 的文件 size 字节
- fwrite <fdnum> std：从屏幕写入句柄为 fdnum 的文件 size 字节
- fread <fdnum> <outfile> <size>：从句柄为 fdnum 的文件读取 size 字节，输出到 outfile 文件
- fread <fdnum> std <size>：从句柄为 fdnum 的文件读取 size 字节，输出到屏幕
- fseek <fdnum> <step> begin：句柄为 fdnum 的文件指针从开头偏移 step
- fseek <fdnum> <step> cur：句柄为 fdnum 的文件指针从当前位置偏移 step
- fseek <fdnum> <step> end：句柄为 fdnum 的文件指针从结尾偏移 step
- fclose <fdnum>：关闭文件句柄为 fdnum 的文件
- fdelete <filename>：删除文件名为 filename 的文件或者文件夹
- exit：退出文件系统
- fsave：保存文件系统（测试命令）

（2）磁盘文件结构

- 实现 SuperBlock 结构，超级块，管理整个文件系统
- SuperBlock 的索引结构设计（成组链接法）
- 实现 inode 结构，管理对应的文件或者目录
- Inode 结构索引结构设计（多级索引）
- 文件（夹）的分配与回收算法

（3）目录结构

- 目录文件的结构
- 目录检索算法的设计与实现
- 目录结构增、删、改的设计与实现

（4）文件打开结构

- 文件打开结构的设计：内存 Inode 节点，File 结构，进程打开文件表
- 内存 Inode 节点的分配与回收
- 文件打开过程
- 文件关闭过程

（5）文件系统实现

- 文件读写操作
- 文件删除操作
- 文件创建操作

（6）高速缓存结构

- 缓存控制块的设计
- 缓存队列的设计及分配和回收算法
- 借助缓存实现对一级文件的读写操作的流程

二，概要设计

2.1 任务分解

类 UNIX 二级文件系统的实现可以分为以下几个部分：

- SuperBlock 模块：
  实现 SuperBlock 数据结构。实现成组链接法分配空闲 block 数据块。维护一个全局 superblock。
- 目录管理模块：
  提供创建目录、删除目录、进入目录、获取当前目录文件列表等操作的相关接口
- 文件管理模块：
  主要负责文件相关的一系列操作。提供创建文件、删除文件、打开文件、关闭文件、写文件、读文件、更改文件指针等操作的相关接口，并且涉及空闲 block 的分配等。
- 高速缓存模块：
  涉及缓存队列的设计及分配和回收，并且借助缓存实现对一级文件的读写操作。
- 顶层模块：
  主要负责用户输入信息处理，将用户输入命令转化为函数操作，并且进行信息输等。

2.2 数据结构定义

（1）SuperBlock：超级块

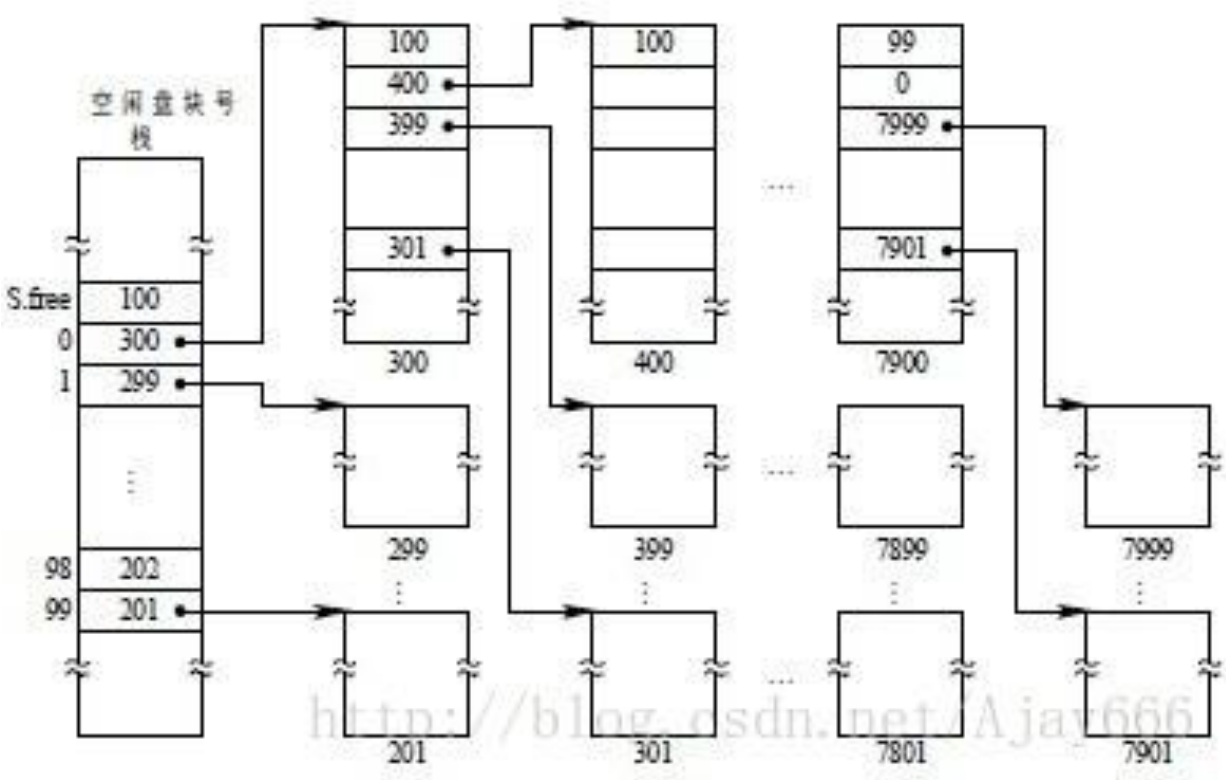
SuperBlock 结构处于文件卷的 0#扇区和 1#扇区，大小为 2048 字节。

数据成员：

- s_isize：inode 区占用的盘块数。Inode 区占用 2#扇区到 59#扇区，占用盘块数为 58 块
- s_fsize：file 部分占用的盘块数。文件部分占用 60#扇区到 16383#扇区。整个文件夹占用 16384 个数据块，即 8MB。
- s_nfree：直接管理的空闲盘块数量。
- s_free：直接管理的空闲盘块索引表，通过成组链接法管理。
- s_ninode：直接管理的空闲外存 Inode 数量。
- s_inode：直接管理的空闲外存 Inode 索引表，通过成组链接法管理。
- padding：填充内存，将 superblock 填充至 1024 字节。

```
1. class SuperBlock
2. {
3.  /* Functions */
4. public:
    int findFreeFile();//成组链接法找空的file块，返回file块号（0开始）
5.  int findFreeInode();
6.
7.  /* Members */
8. public:
9.  int  s_isize;  /* 外存Inode区占用的盘块数 */
10. int  s_fsize;  /* file部分盘块总数 */
11.
```

```
12. int  s_nfree;  /* 直接管理的空闲盘块数量 */
13. int  s_free[FREE_INDEX_NUM]; /* 直接管理的空闲盘块索引表 从0开始计数 */
14.
15. int  s_ninode;  /* 直接管理的空闲外存Inode 数量 */
16. int  s_inode[FREE_INDEX_NUM]; /* 直接管理的空闲外存Inode 索引表 */
17.
18. int  padding[52 + (FREE_INDEX_NUM - 100)]; /* 填充使SuperBlock 块大小等
    于1024 字节，占据2 个扇区 */
19.};
```



成组链接法管理空闲盘块的分配：

SuperBlock 结构存储了直接管理的空闲盘块列表，大小为 100，然而 block 扇区数在于 60#~16383#，100 显然不够存储，所以需要成组链接法进行管理。直接管理的空闲盘块列表 s_free，其中 s_free[1]~s_free[99]存储空闲 block 的盘块号。s_free[0]存储链接块的盘块号。链接块的结构与 s_free 一致，一样是[1]~[99]存储空闲 block 盘块号,[0]存储下一链接块的地址，直到存储结束，[0]存储 0。

空闲盘块需要 164 个链接块存储，存在 61#-224#扇区，空闲 inode 需要 4 个链接块存储，存在 225#~228#扇区。

根据我的数据结构，查找空闲的块号时，我需要先在 superblock 中查找，如果 s_nfree 大于 0，则直接在直接索引块中取即可，若是 s_nfree=0，则需要根据 s_free[0]去找到下一个链接块，以此类推。若是当 s_free[0] = 0（最后一个链接块）且此链接块也分配完了，则抛出错误信息，即空间不足。

（2）DiskInode：一个目录/文件对应一个 inode

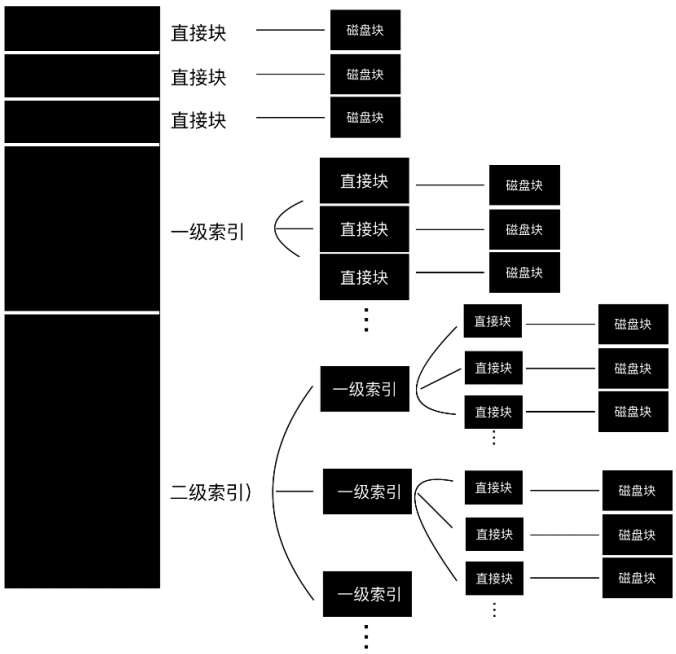
DiskInode 是单个 inode 数据结构，而 inode 区是紧跟在 superblock 后面的 2#扇区到 59#扇区。一个 DiskInode 数据结构占 64 字节，一个扇区 512 字节，共可以存 8 个 inode。

数据成员：

- d_mode：文件类型，有 FILE（文件）与 DIRECTORY（目录）两种

- f_flag：文件权限，有 FREAD（只读）与 FWRITE（可写）两种
- f_offset：文件读写指针，从该位置开始读/写
- d_size：文件大小
- d_addr：用于文件逻辑块号和物理块号转换的基本索引表
- d_atime：最后访问时间

```
1. class DiskInode
2. {
3. public:
4.  enum InodeMode {
5.   FILE = 0x1,//是文件
6.   DIRECTORY = 0x2//是目录
7.  };
8.  enum FileFlags
9.  {
10.  FREAD = 0x1,    /* 读请求类型 */
11.  FWRITE = 0x2    /* 写请求类型 */
12. };
13.
14. unsigned int d_mode; /* 状态的标志位，定义见 enum InodeMode */
15.
16. unsigned int f_flag;  /* 对打开文件的读、写操作要求 */
17. int  f_offset;   /* 文件读写位置指针 */
18.
19. int  d_size;   /* 文件大小，字节为单位 */
20. int  d_addr[10];  /* 用于文件逻辑块好和物理块好转换的基本索引表 */
21.
22. time_t  d_atime;  /* 最后访问时间 */
23. //time_t 占用 8 字节
24.};
```



Inode 多级索引：

d_addr 是 inode 数据结构中的索引部分，用于查找本文件的内容处在数据区的哪一个扇区。其中 d_addr[0]~[5]是直接块，直接存储对应的扇区的编号，可以存储 0~(6*512＝3KB)。d_addr[6]~[7]是一级索引，存储的是一页间接索引（也在数据区）的扇区号，对应的间接索引可以存储 128 个扇区号，可以存储(128*2+6)*512＝131KB 数据。d_addr[8]~[9]是二级索引，存储的是一页一级索引（也在数据区）的扇区号，对应的一级索引可以存储 128 个间接索引的扇区号，对应的间接索引也可以存储 128 个直接块扇区号，可以存储(128*128*2+128*2+6)*512 = 16515KB 数据。

（3）File：文件

File 是数据区的数据结构，用于存储文件 FILE 类型的数据。内容即为一个大小为 BLOCK_SIZE 的数组（512 字节）。若本数据块是该文件的第一个数据块，则前 FILENAME（20）个字节是文件名。

```
1. class File
2. {
3. public:
4.  char content[BLOCK_SIZE];//addr[0]对应的前 FILENAME 个 char 是名字
5. };
```

装

（4）Directory：目录

Directory 也是数据区的数据结构，用于存储目录的数据。一个目录有且仅有一个数据块。目录的前两个成员是自身(.)与父亲(..)。

数据成员：

- name：目录名
- d_inodenumber：子文件的 inode 号
- d_filename：子文件名，与 d_inodenumber 一一对应

订

```
1. class Directory
2. {
3. public:
4.  char name[FILENAME_LENGTH];//目录名字
5.  int d_inodenumber[SUBDIRECTORY_NUM];  //子目录 Inode 号 //-1
6.  char d_filename[SUBDIRECTORY_NUM][FILENAME_LENGTH];  //子目录文件
   名 //\0
7.  //总共大小 500 字节
8. };
```

线

（5）CacheBlock：缓存控制块

CacheBlock 缓存控制块，描述每一个高速缓存的状态，并且存储了对应缓存块的内容。

数据成员：

- flags：缓存控制块标志位，有 CB_DONE 操作结束 与 CB_DELWRI 需要延迟写
- forw：缓存块循环链表指针，指向前一个缓存控制块。
- back：缓存块循环链表指针，指向后一个缓存控制块。
- type：cache 存放的数据的类型，有 INODE 与 BLOCK 两种
- wcount：需要写回的字节数
- buffer：缓冲区数组
- blkno：写回的磁盘逻辑块号
- no：缓存控制块序号。

```
1. class CacheBlock
```

```
2. {
3. public:
4.  CacheBlock();
5.  void reset();
6.
7. public:
8.  static const int BUFFER_SIZE = 512;      //缓冲区大小 512 字节
9.  enum CacheFlag
10. {
11.  CB_DONE = 0x1,     //I/O 操作结束
12.  CB_DELWRI = 0x2    //延迟写
13. };
14. enum CacheType
15. {
16.  INODE = 0x1,
17.  BLOCK = 0x2
18. };
19. unsigned int flags;    //缓存控制块标志位
20.
21. CacheBlock* forw;
22. CacheBlock* back;
23.
24. int type;      //cahce 存放的模块类型
25. int wcount;      //需传送的字节数
26. char buffer[BUFFER_SIZE]; //缓冲区数组
27. int blkno;      //磁盘逻辑块号
28. int no;
29.};
```

（6）CacheList：缓存队列

CacheList 缓存队列。

数据成员：

- bufferList：自由缓存队列控制块循环链表，查找空缓存快在这里找
- nBuffer：缓存控制块数组，所有的缓存块都存在这里

方法：

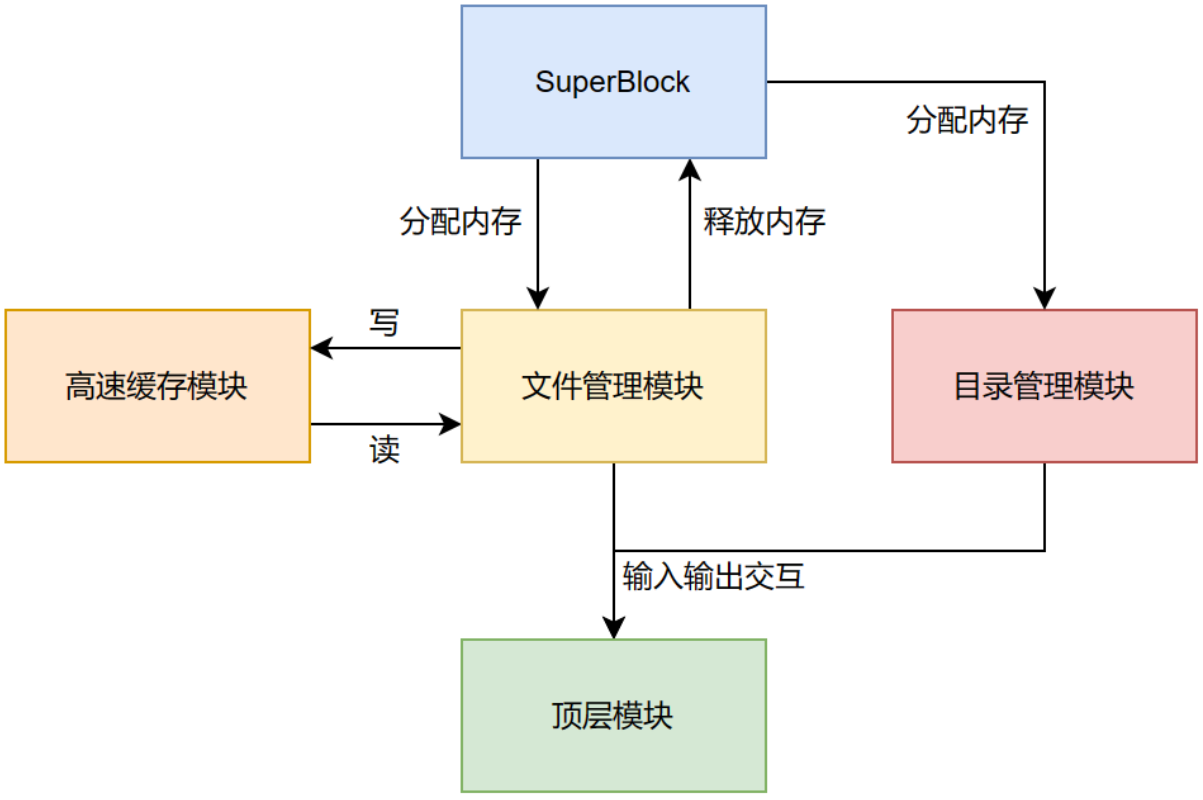
- CacheList()：构造函数，初始化循环链表。
- ~CacheList()：析构函数。
- initList()：初始化循环链表
- cacheIn(CacheBlock* pb)：LRU 算法插入缓存块，插入到循环链表末尾
- cacheOut(CacheBlock* pb)：LRU 算法取出缓存块，从循环链表开头取出
- findCache(int blkno, int type)：寻找要写入 blkno 的 cache 是否存在
- writeCache()：写入 cache
- readCache(CacheBlock* pb, char* content)：读 cache
- freeCache(CacheBlock* pb)：释放缓存块
- writeBack()：延迟写，脏缓存写回文件卷

```
1. class CacheList
2. {
3. public:
```

```
4.  static const int NBUF = 100;              //缓存控制块 缓冲区的数量
5.  static const int BUFFER_SIZE = 512;       //缓冲区大小 512 字节
6.
7.  CacheList();
8.  ~CacheList();
9.  void initList();  //init
10. void cacheIn(CacheBlock* pb); //插入缓存块
11. void cacheOut(CacheBlock* pb); //缓存块 out
12. CacheBlock* findCache(int blkno, int type);//看 blkno 是否存在缓存块中了
13. void writeCache(int blkno, int type, const char* content, int length);
14. int readCache(CacheBlock* pb, char* content);
15. void freeCache(CacheBlock* pb);//释放缓存块，放回队列中
16. void writeBack();//脏缓存全部写回 img 中
17.
18.private:
19. CacheBlock* bufferList;          //自由缓存队列控制块，查找空缓存快在这里找
20. CacheBlock nBuffer[NBUF];        //缓存控制块数组，所有的缓存块，查找是否在
    cache 的时候在这里找
21.
22.};
```

装 订 线

2.3 模块间的调用关系



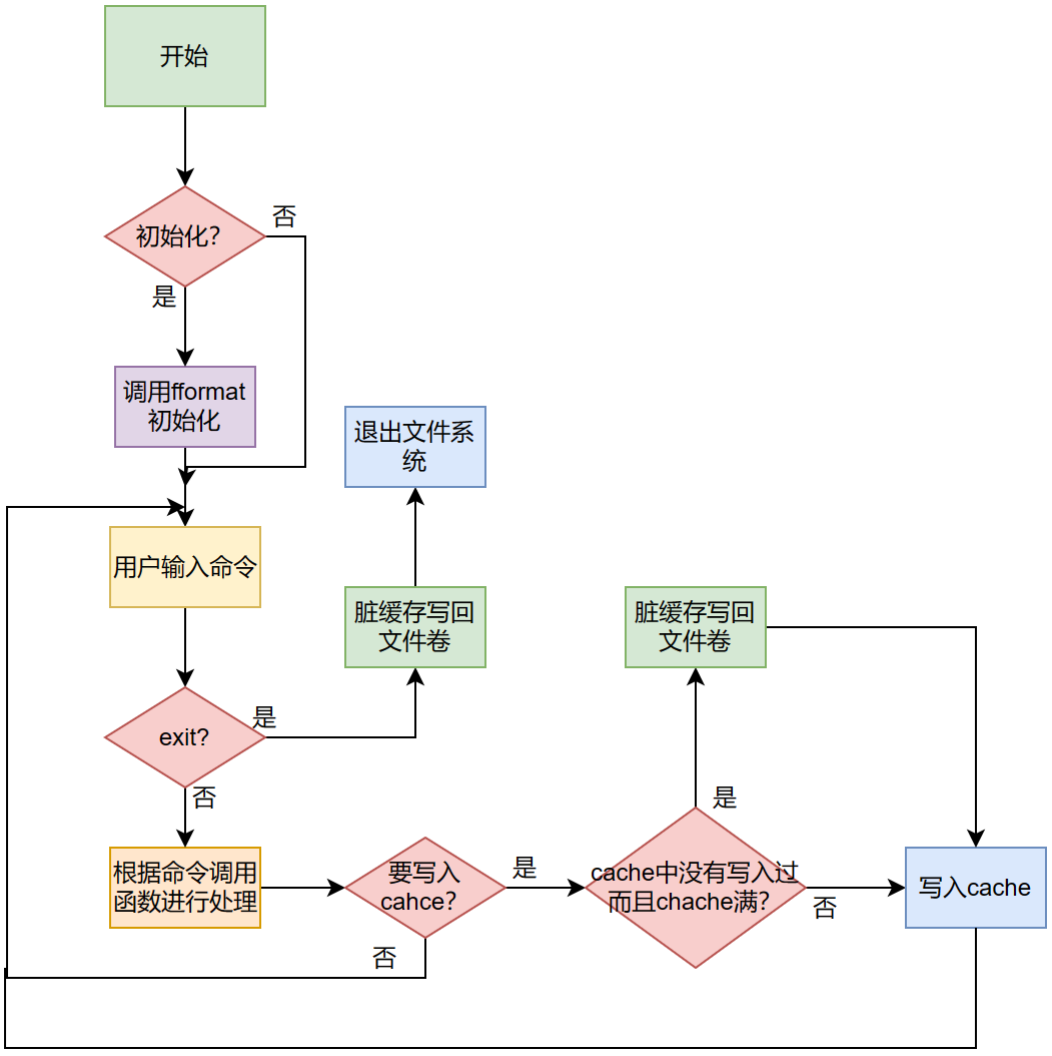
2.4 算法说明

整个文件系统的算法流程如下图：

运行程序，用户选择是否进行初始化，如果要初始化，则调用 fformat 进行初始化，否则从文件卷读入全局 superblock 和根目录，进入用户输入阶段。

用户输入阶段，用户输入具体的命令，解码执行。看是否要调用 cache 模块，如果要调用 cache 写入，则看 cache 是否有对应 blkno 的缓存块，有就直接写入覆盖，否则就要申请一个 cache 缓存块来存，若是 cache 已满则把脏缓存全部写入文件卷，再申请一个 cache 缓存块。若不需要调用 cache 模块则直接回到用户输入阶段，循环。

若用户调用 exit 函数，则将脏缓存全部写回文件卷，并且写回全局 superblock 和目录 directory，退出文件系统。



装 订 线

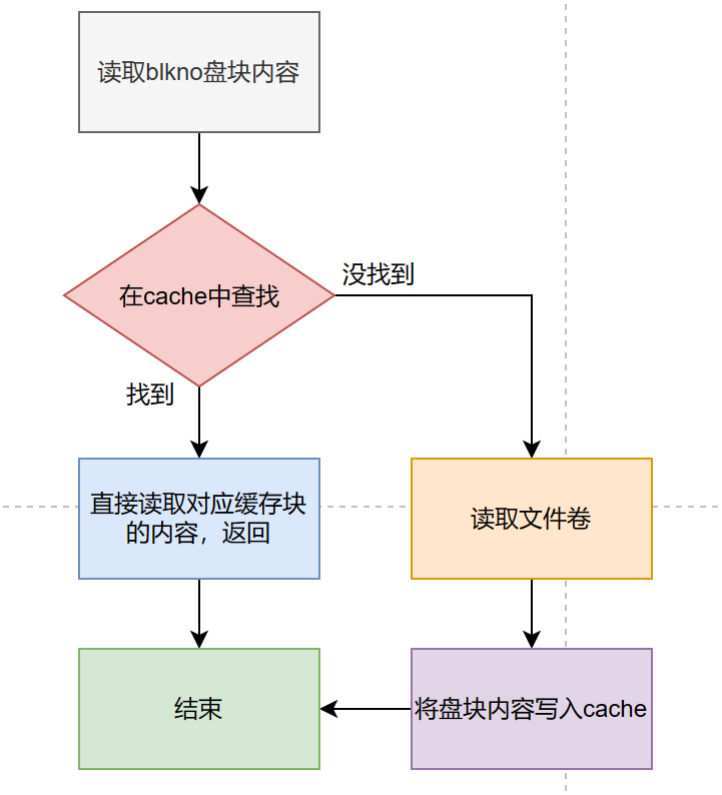
三，详细设计

3.1 一些重要基本函数

3.1.1 getblk 读取 blkno 块号 block 的内容

```
void getblk(const int blkno, int type, char* content, int length);
```

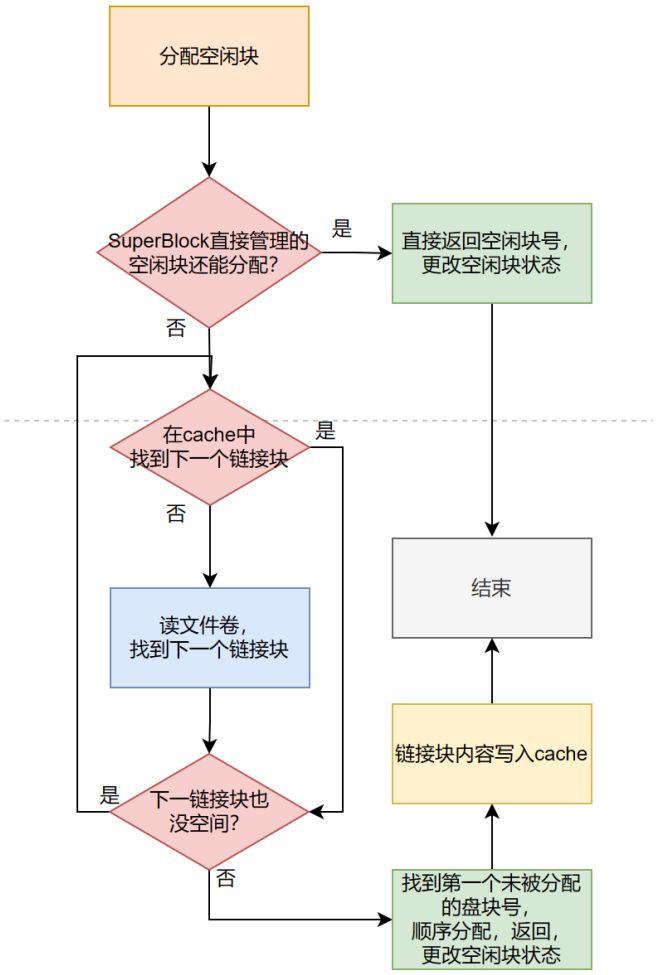
在涉及读写的过程中，很多时候需要读相应的 blkno 块号的盘块的内容，这个时候我们要先去 cache 里寻找，未果再去读文件卷。这个操作经常需要施行，以下是其流程图：



3.2 文件系统模块

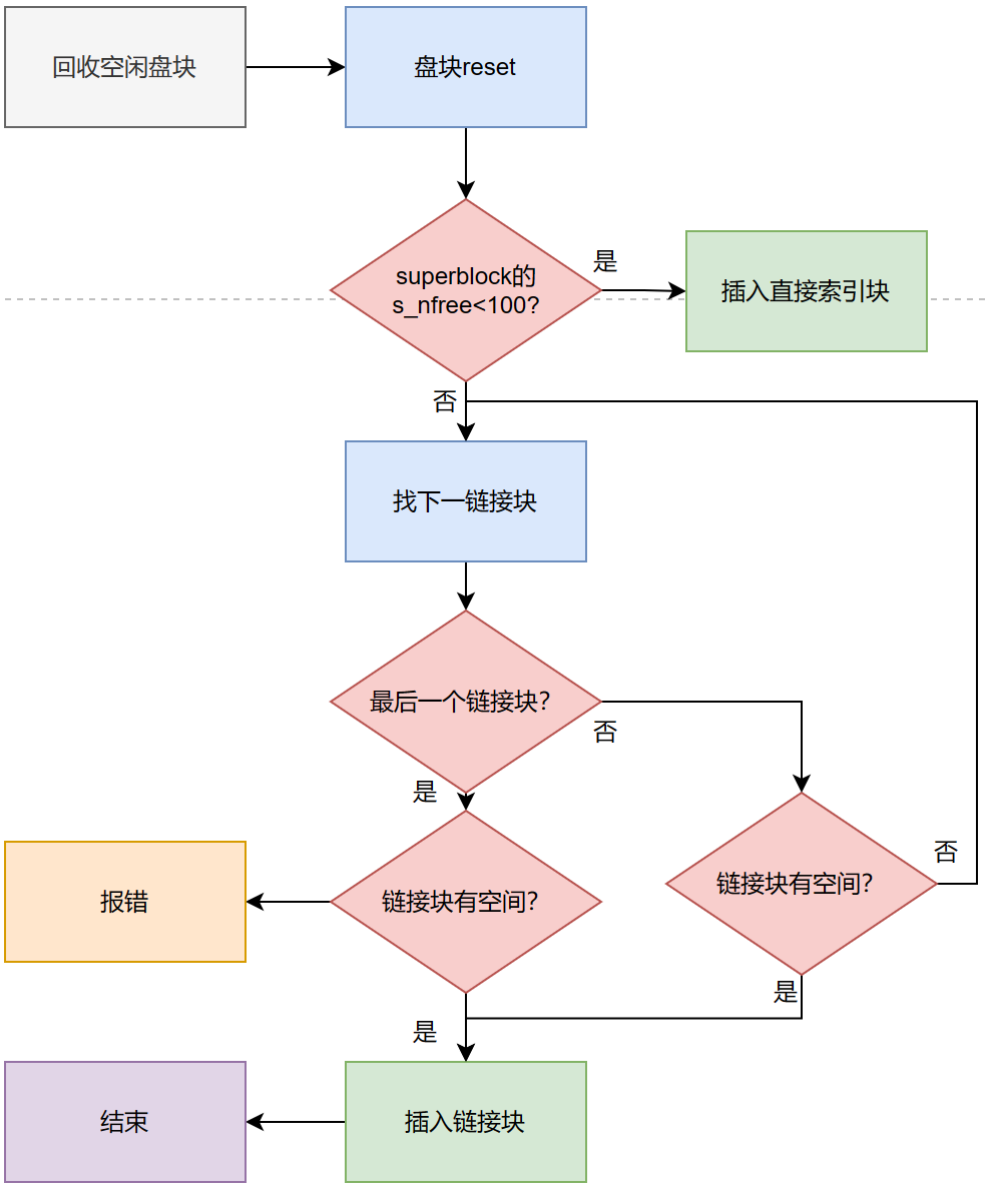
3.2.1 SuperBlock 空闲盘块分配

在进行写操作的时候经常需要分配一个新的空闲块，这个时候就需要 superblock 返回一个空闲盘块的盘块号。以下是流程图：



3.2.2 回收一个空闲盘块

当运行删除操作的时候，会涉及到空闲盘块的回收操作，将盘块 reset 清空，这个时候会涉及到空闲盘块的插入回 superblock 索引块的操作，又因为我们采用的是成组链接法，所以要根据成组链接法去回收。我们首先考虑 superblock 的直接索引块，能插入就直接插入，因为我们分配的时候也是先从直接索引块取出的，所以可以省去一些不必要的 IO 操作；若是直接索引块满了，则需要根据成组链接法找到下一个链接块，重复插入操作。若是全部索引块都是满的（一般不会出现这种情况），则报错。以下是这个算法的流程图：



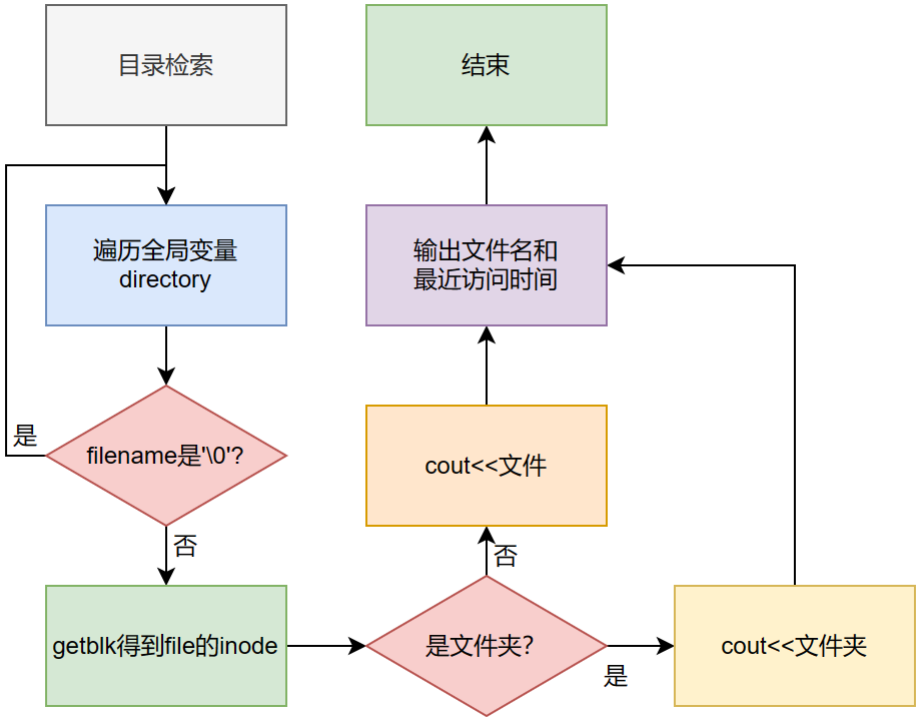
3.3 目录管理模块

我的文件系统中会将当前目录存在内存中，即我会有一个全局变量 directory，保存当前目录的 block 块信息，而我的目录算法将基于这个全局变量进行操作。

3.3.1 目录检索

```
void ls();
```

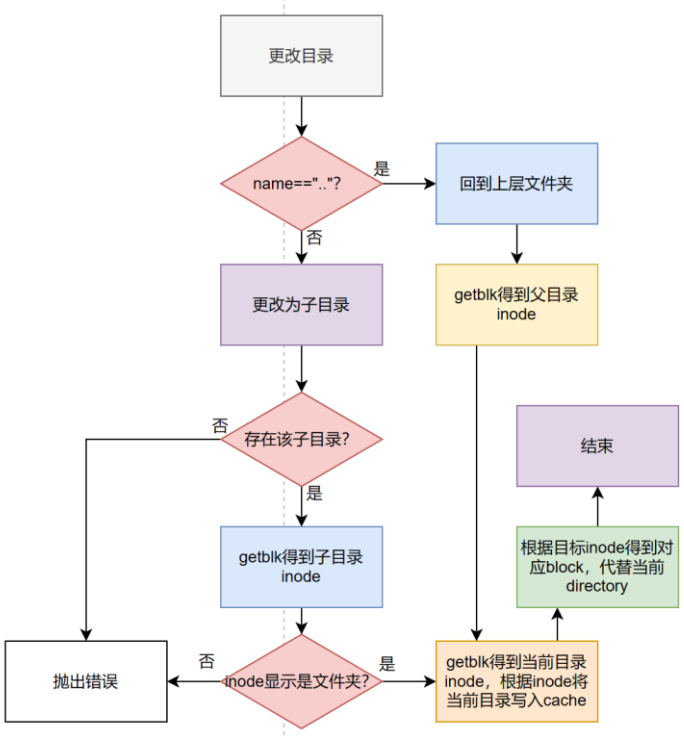
目录检索算法将直接遍历 directory 的 filename 结构，得到子文件（夹）的 inode 盘块号，得到 inode 信息，输出文件类型，文件名和最近访问时间。以下是流程图：



3.3.2 更改目录

```
void cd(const char* name);
```

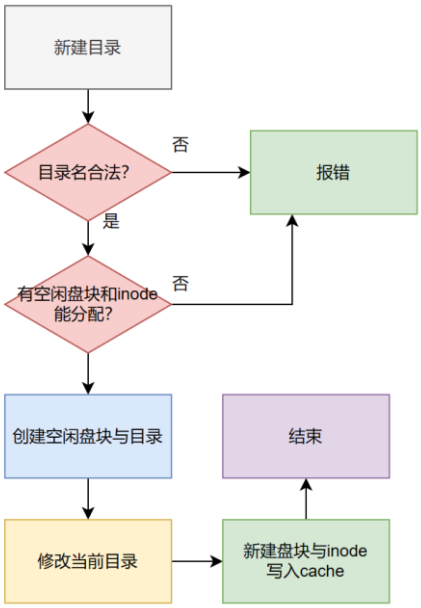
更改目录是一个很常用的操作。可以分为进入下层目录和回到上层目录两种情况。以下是其流程图：



3.3.3 新建目录

```
void mkdir(const char* name);
```

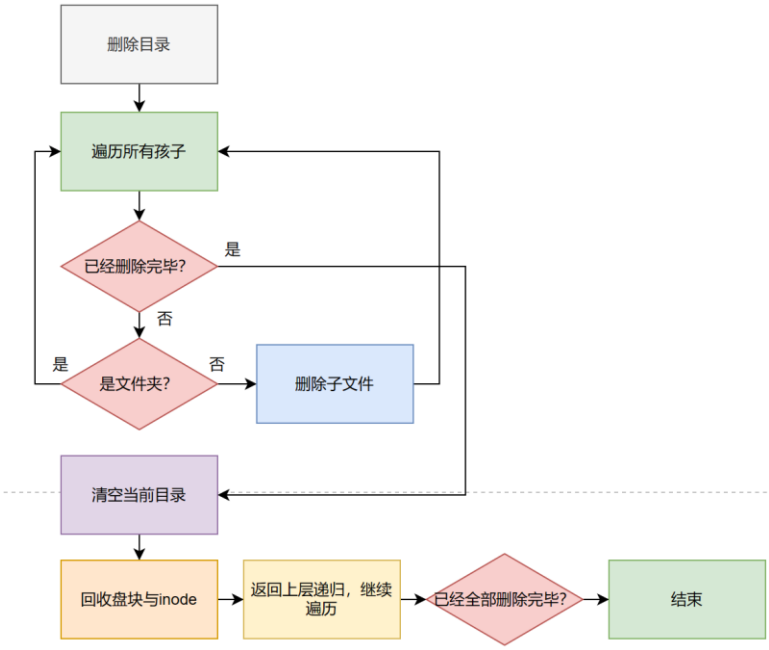
创建目录成功的前提是目录名合法且有空闲的 block 和 inode。以下是其流程图：



3.3.4 删除目录

```
inline void delete_directory();
```

删除目录是一个非常复杂的过程，因为删除的目录可能会有子目录和子文件，若是仅仅删除了当前目录而不去管子文件，这些字文件就丢失了，但是空间还占用着，显然不合理。所以本文件系统采用递归原理，先删除子文件和子目录，再将本目录删除。以下是流程图：流程图中删除文件的部分并没有详细展开，该部分将在文件操作部分详细阐述；流程图中回收盘块和 inode 部分在文件系统模块 3.2.2 回收空闲盘块部分讲述。



3.4 文件操作模块

文件操作模块主要包括文件的读写，新建，删除，打开，关闭等操作，是本文件系统的主要交互方向。

3.4.1 文件新建

新建文件与新建文件夹类似，进行基本的判断（文件名是否合法，能否申请空闲的盘块与 inode）之后，新建相应的盘块与 inode，修改当前目录，最后将新建的盘块与 inode 写入 cache。

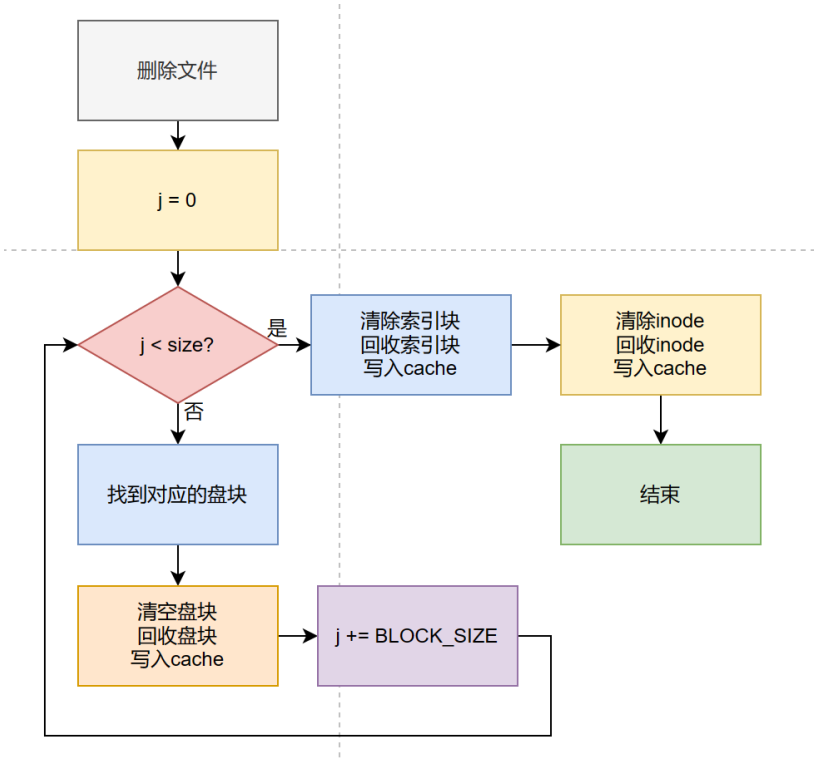
3.4.2 文件打开与关闭

文件打开与关闭我定义了一个数组用来存打开文件的 inode 编号，以此来记录文件的打开与关闭操作。当文件打开，相应文件的 inode 编号写入 open_file 数组，当文件关闭，相应的 inode 也要剔除出 open_file 数组。可以说，文件的打开与关闭操作就是对这个数组进行。

3.4.3 文件删除

```
inline void delete_file(const DiskInode inode, int i);
```

删除文件操作主要是删除两个部分：一是 inode 部分，二是盘块部分。Inode 部分的删除很简单，但是盘块部分的删除涉及文件的二级索引结构，需要根据文件的大小，找到所有的盘块，将其删除，同时也要将所有的索引块一并删除。以下是其流程图：

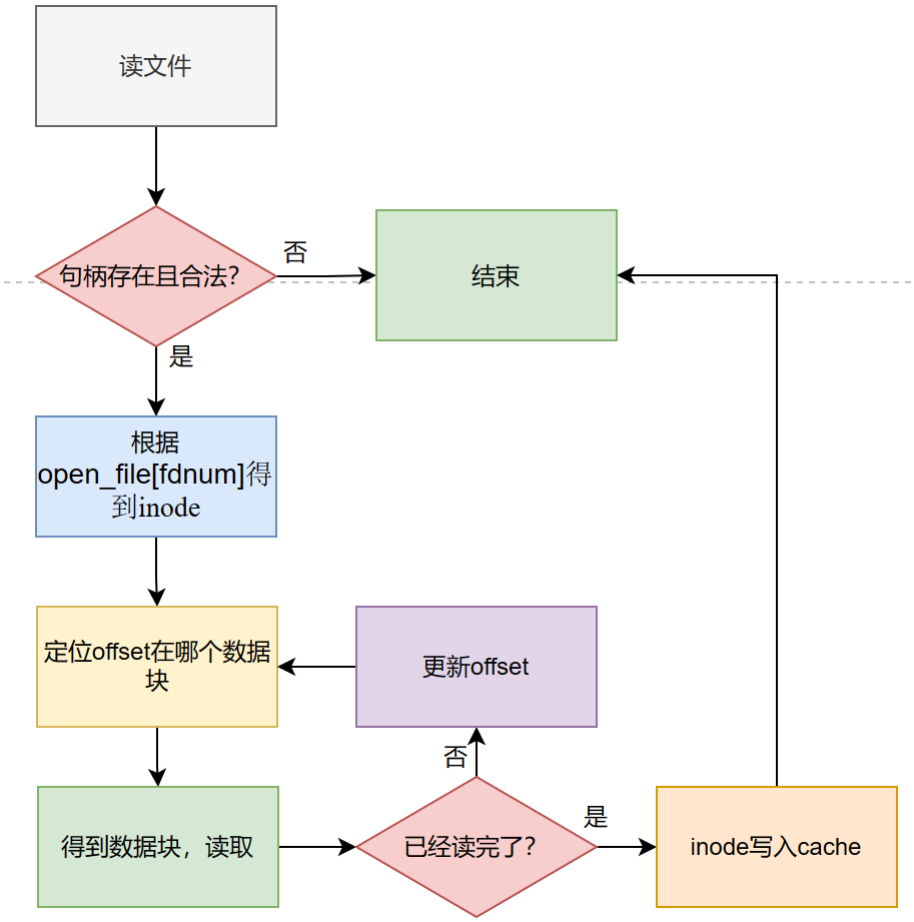


3.4.4 文件读写

3.4.4.1 文件读

```
string fread(int fdnum, int size);
```

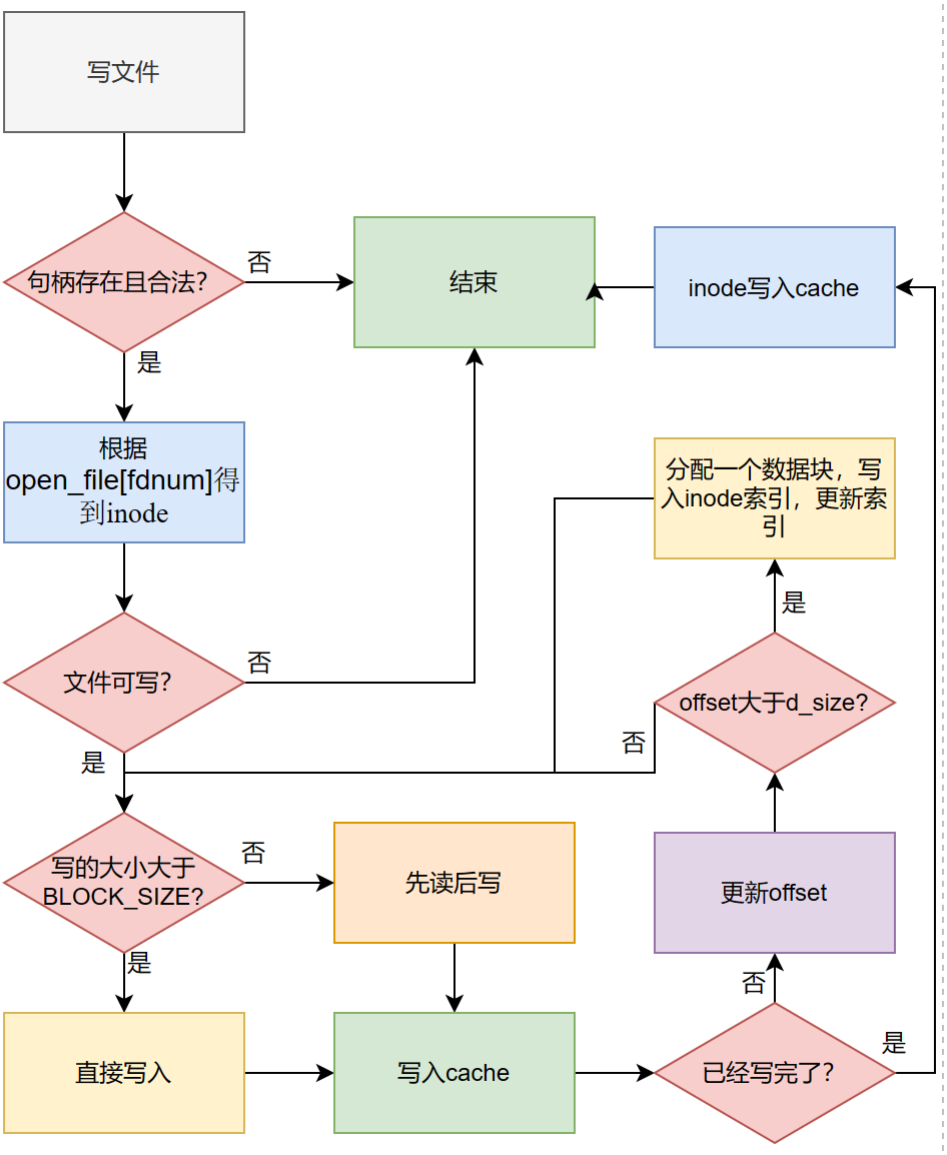
读文件首先判断输入的文件句柄 fdnum 是否合法，且 open_file 列表中是否存在 fdnum 句柄（即文件是否已打开）。然后根据 open_file[fdnum]得到文件的 inode，然后根据 offset 开始读文件。首先定位本 offset 在哪一个数据块，然后根据 size 读数据块，若还没读完，则更新 offset，继续定位数据块，继续读，以此循环，直到读了 size 个字节。最后全部读完后，将 inode 写入 cache，返回。下面是其流程图：



3.4.4.2 文件写

```
void fwrite(int fdnum,const char* content_);
```

文件写的过程与文件读形式上非常相似，但是会更加复杂一点。同样的，先判断句柄（fdnum）合理性，然后根据 open_file[fdnum]得到文件的 inode，然后判断文件是否可写。若可写，则开始进行文件写。定位 offset 在哪个数据块，如果要写的字节数大于 BLOCK_SIZE，则直接写，然后写入 cache，否则要先读后写。然后更新 offset，进行循环。如果 offset 大于等于文件大小，则要新申请一个数据块用来写，并且将数据块加入 inode 索引。最后将 inode 写入 cache，结束。下面是其流程图：



3.4.5 更改文件指针

```
void fseek(int fdnum, int dis, int type);
```

同样的，更改文件指针首先需要判断文件句柄 fdnum 的合法性。fseek 提供了三个接口，分别是 BEG（文件头），CUR（此刻位置），END（文件尾），根据用户选择的模式，对应进行修改文件的 offset 指针即可。

3.5 高速缓存模块

高速缓存模块实现 cache，主要实现了缓存块的分配，回收，读写操作等。

3.5.1 cache 分配

```
void CacheList::cacheOut(CacheBlock* pb);
```

cache 分配和回收采用的是 LRU 算法。我的缓存块采用的是一个循环链表存储。当我分配缓存块时，要取循环链表中的第一项来分配，并且 pop 出循环链表。

3.5.2 cache 回收

```
void CacheList::writeBack();
```

当文件系统 exit，或者是没有多余的缓存块来分配了，则需要回收 cache。需要遍历缓存块，找到所有脏缓存，并且将缓存内容写入文件卷，然后释放缓存块。释放缓存块 reset 之后，需要插入到循环链表中，根据 LRU 算法，插入到循环链表的末尾。

3.5.3 cache 读

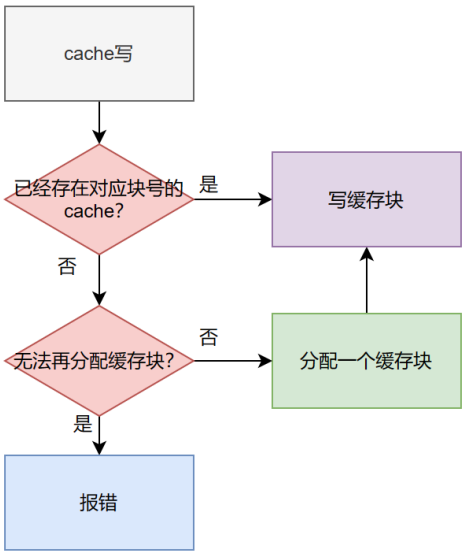
```
int CacheList::readCache(CacheBlock* pb, char* content);
```

cache 读即直接将对应 cache pb 的内容读到 content 中即可。

3.5.4 cache 写

```
void CacheList::writeCache(int blkno,int type,const char* content,int length);
```

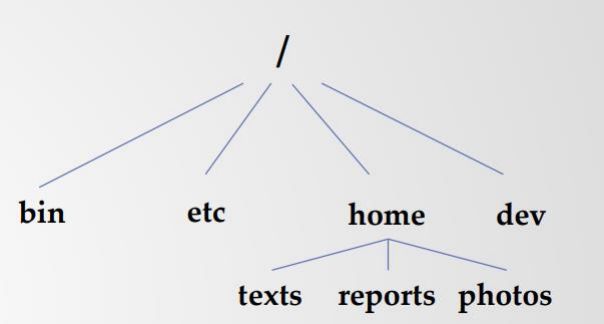
cache 写首先需要判断是否有 cache 已经写入了块号 blkno。如果已经存在了，则直接写入对应 cache 即可，并且更新缓存块标记。如果不存在，则需要申请一个 cache 缓存块。下面是其流程图：



四，运行结果分析

通过命令行方式完成下列操作：

- 格式化文件卷；
- 用 mkdir 命令创建子目录，建立如图所示目录结构；
- 把你的课设报告，关于课程设计报告的 ReadMe.txt 和一张图片存进这个文件系统，分别放在/home/ texts ，/home/reports 和/home/photos 文件夹；

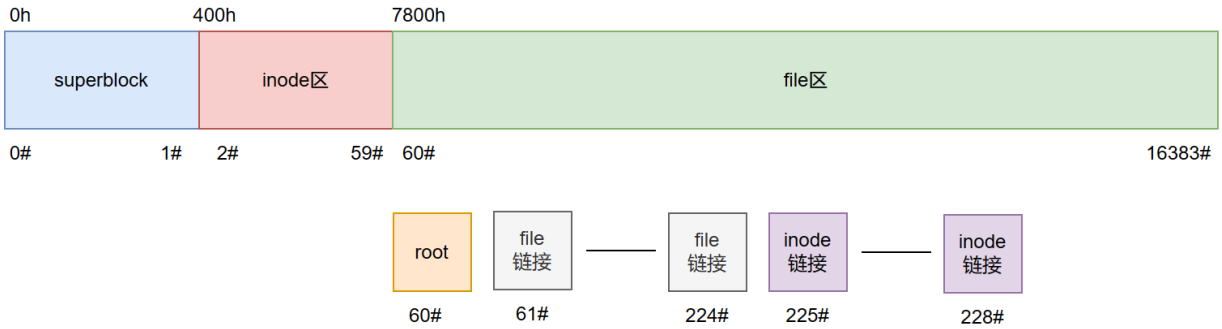


通过命令行方式测试：

- 新建文件/test/Jerry，打开该文件，任意写入 800 个字节；
- 将文件读写指针定位到第 500 字节，读出 500 个字节到字符串 abc。
- 将 abc 写回文件。

4.1 格式化文件卷

根据我们前面的描述，我们的文件卷大小总共有 8MB。其中 0#与 1#是 superblock 区，起始地址是 0h(0#)，inode 区是 2# - 59#，起始地址是 400h(2#)，file 区是 60#到 16383#，起始地址是 7800h(60#)。而初始化后 inode 区应有一个 inode，为 root 目录的 inode，占用 2#的第一个 inode 块。File 区 60#扇区存放 root 目录，61#到 224#扇区存放 file 空闲块的链接索引，起始地址 7a00h(61#)，225# - 228#扇区存放 inode 空闲块链接索引，起始地址 1c200h(225#)。而且根据 inode 和 file 块的使用情况，inode 已经使用了 1 块，file 已经使用了 169 块，根据空闲块分配算法，superblock 中的空闲块索引已经全部占用完毕，61#中的空闲块索引占用了 70 块。如下图所示。



运行程序，进行初始化

```
********************************************************************************
是否对文件系统进行初始化？[y/n]
y

[@ root ]$
```

查看文件卷，是否符合我们的推算结果

（1）400h 处是 inode 区，且占用一个 inode 块

```
00000400h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000410h: 00 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ....烫烫烫烫烫烫
00000420h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
00000430h: CC CC CC CC CC CC CC CC BB CF 48 66 00 00 00 00 ; 烫烫烫烫幌Hf....
00000440h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000450h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000460h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000470h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000480h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000490h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

（2）7800h 处是 file 区，存放 root 目录

```
00007800h: 72 6F 6F 74 00 CC CC CC CC CC CC CC CC CC CC CC ; root.烫烫烫烫烫?
00007810h: CC CC CC CC 00 00 00 00 00 00 00 00 FF FF FF FF ; 烫烫........
00007820h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00007830h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00007840h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00007850h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00007860h: FF FF FF FF 2E 00 CC CC CC CC CC CC CC CC CC CC ;         ..烫烫烫烫烫
00007870h: CC CC CC CC CC CC CC CC 2E 2E 00 CC CC CC CC CC ; 烫烫烫烫...烫烫?
00007880h: CC CC CC CC CC CC CC CC CC CC CC CC 00 CC CC CC ; 烫烫烫烫烫烫 烫?
```

（3）File 链接块使用了 169 块，则 superblock 中的空闲块索引已经全部占用完毕，61#中的空闲块索引占用了 70 块。

```
           0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 3A 00 00 00 C4 3F 00 00 00 00 00 00 01 00 00 00 ; :...?..........
00000010h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000020h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000030h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000040h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000050h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000060h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000070h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000080h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000090h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000a0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000b0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000c0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000d0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000e0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000f0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000100h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000110h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000120h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000130h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

装 订 线

```
00007a00h: 63 00 00 00 02 00 00 00 C5 00 00 00 C4 00 00 00 ; c.......?..?..
00007a10h: C3 00 00 00 C2 00 00 00 C1 00 00 00 C0 00 00 00 ; ?..?..?..?..
00007a20h: BF 00 00 00 BE 00 00 00 BD 00 00 00 BC 00 00 00 ; ?..?..?..?..
00007a30h: BB 00 00 00 BA 00 00 00 B9 00 00 00 B8 00 00 00 ; ?..?..?..?..
00007a40h: B7 00 00 00 B6 00 00 00 B5 00 00 00 B4 00 00 00 ; ?..?..?..?..
00007a50h: B3 00 00 00 B2 00 00 00 B1 00 00 00 B0 00 00 00 ; ?..?..?..?..
00007a60h: AF 00 00 00 AE 00 00 00 AD 00 00 00 AC 00 00 00 ; ?..?..?..?..
00007a70h: AB 00 00 00 AA 00 00 00 A9 00 00 00 FF FF FF FF ; ?..?..?..
00007a80h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00007a90h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00007aa0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00007ab0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
```

初始化 img 大小

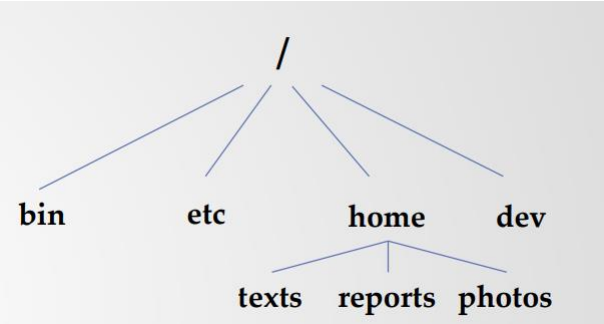
myDisk.img	2024/5/19 1:29	光盘映像文件	115 KB

装

订

线

4.2 用 mkdir 命令创建子目录，建立如图所示目录结构；



运行程序建立。

```
[@ root ]$ mkdir bin
新建文件夹 bin 完成

[@ root ]$ mkdir etc
新建文件夹 etc 完成

[@ root ]$ mkdir home
新建文件夹 home 完成

[@ root ]$ mkdir dev
新建文件夹 dev 完成

[@ root ]$ cd home

[@ home ]$ mkdir texts
新建文件夹 texts 完成

[@ home ]$ mkdir reports
新建文件夹 reports 完成

[@ home ]$ mkdir photos
新建文件夹 photos 完成
```

```
[@ home ]$ ls
当前目录: home
子目录:
************************************************
  文件类型     文件名     最后访问时间
************************************************
  文件夹       texts      Sun May 19 00:33:43 2024

  文件夹       reports      Sun May 19 00:33:49 2024

  文件夹       photos      Sun May 19 00:33:57 2024


[@ home ]$ cd ..

[@ root ]$ ls
当前目录: root
子目录:
************************************************
  文件类型     文件名     最后访问时间
************************************************
  文件夹       bin      Sun May 19 00:33:11 2024

  文件夹       etc      Sun May 19 00:33:16 2024

  文件夹       home       Sun May 19 00:33:20 2024

  文件夹       dev      Sun May 19 00:33:25 2024
```

装

此时 inode 应有 8 个。

订

线

```
           0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
000003f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000400h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000410h: 00 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ....烫烫烫烫烫烫
00000420h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
00000430h: CC CC CC CC CC CC CC CC BB CF 48 66 00 00 00 00 ; 烫烫烫烫幌Hf....
00000440h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000450h: A9 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ?..烫烫烫烫烫烫
00000460h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
00000470h: CC CC CC CC CC CC CC CC 47 D8 48 66 00 00 00 00 ; 烫烫烫烫G碼f....
00000480h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000490h: AA 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ?..烫烫烫烫烫烫
000004a0h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
000004b0h: CC CC CC CC CC CC CC CC 4C D8 48 66 00 00 00 00 ; 烫烫烫烫L碼f....
000004c0h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000004d0h: AB 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ?..烫烫烫烫烫烫
000004e0h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
000004f0h: CC CC CC CC CC CC CC CC 50 D8 48 66 00 00 00 00 ; 烫烫烫烫P碼f....
00000500h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000510h: AC 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ?..烫烫烫烫烫烫
00000520h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
00000530h: CC CC CC CC CC CC CC CC 55 D8 48 66 00 00 00 00 ; 烫烫烫烫U碼f....
00000540h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000550h: AD 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ?..烫烫烫烫烫烫
00000560h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
00000570h: CC CC CC CC CC CC CC CC 67 D8 48 66 00 00 00 00 ; 烫烫烫烫g碼f....
00000580h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000590h: AE 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ?..烫烫烫烫烫烫
000005a0h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
000005b0h: CC CC CC CC CC CC CC CC 6D D8 48 66 00 00 00 00 ; 烫烫烫烫m碼f....
000005c0h: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000005d0h: AF 00 00 00 CC CC CC CC CC CC CC CC CC CC CC CC ; ?..烫烫烫烫烫烫
000005e0h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫
000005f0h: CC CC CC CC CC CC CC CC 75 D8 48 66 00 00 00 00 ; 烫烫烫烫u碼f....
00000600h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000610h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

合理。

4.3 把你的课设报告，关于课程设计报告的 ReadMe.txt 和一张图片存进这个文件系统，分别放在/home/ texts ，/home/reports 和/home/photos 文件夹

（1）在/home/ texts 目录下新建一个文件 readme，放入 readme.txt

```
[@ root ]$ cd home

[@ home ]$ cd texts

[@ texts ]$ fcreate readme
新建文件 readme 完成

[@ texts ]$ ls
当前目录: texts
子目录:
************************************************
  文件类型      文件名      最后访问时间
************************************************
  文件          readme      Sun May 19 00:49:00 2024


[@ texts ]$ fopen readme
打开文件 readme 成功，文件句柄 0

[@ texts ]$ fwrite 0 Readme.txt
当前文件指针：0，写入字节数：2462
```

查看文件卷，成功将字节数写入，占用 5 个空闲块

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0001d7a0h: CC CC CC CC 00 CC CC CC CC CC CC CC CC CC CC CC ; 烫烫.烫烫烫烫烫烫?
0001d7b0h: CC CC CC CC CC CC CC CC CC 00 CC CC CC CC CC CC ; 烫烫烫烫烫.烫烫烫?
0001d7c0h: CC CC CC CC CC CC CC CC CC CC CC CC CC 00 CC CC ; 烫烫烫烫烫烫烫.烫?
0001d7d0h: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; 烫烫烫烫烫烫烫烫烫
0001d7e0h: 00 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ; .烫烫烫烫烫烫烫烫?
0001d7f0h: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00 ; 烫烫............
0001d800h: E4 B8 80 E3 80 81 E7 A8 8B E5 BA 8F E5 BD A2 E5 ; 涓€鎶仭▼搴忓舰?
0001d810h: BC 8F EF BC 9A 63 6D 64 E5 BD A2 E5 BC 8F 0D 0A ; 纺锛歝md褰㈠纺..
0001d820h: 09 E5 BC 80 E5 8F 91 E5 B9 B3 E5 8F B0 3A 20 57 ; .寮€鍙戝钩鍙? W
0001d830h: 69 6E 64 6F 77 73 20 31 31 0D 0A 09 E5 BC 80 E5 ; indows 11...寮€?
0001d840h: 8F 91 E5 B7 A5 E5 85 B7 EF BC 9A 56 53 32 30 32 ; 崇宸ュ叿锛歏S202
0001d850h: 32 20 78 38 36 0D 0A 09 E9 87 87 E7 94 A8 E5 91 ; 2 x86...閲囩敤鍛
0001d860h: BD E4 BB A4 E8 A1 8C E4 BA A4 E4 BA 92 E6 96 B9 ; 戒护琛屽氦浜掍柟
0001d870h: E6 B3 95 E8 BF 90 E8 A1 8C EF BC 8C 38 30 30 2E ; 娉曞緱琛岃紝800.
0001d880h: 74 78 74 E3 80 81 52 65 61 64 6D 65 2E 74 78 74 ; txt鍚丶eadme.txt
0001d890h: E3 80 81 72 65 70 6F 72 74 2E 70 64 66 E3 80 81 ; 鍚嶱eport.pdf鍚?
0001d8a0h: 70 68 6F 74 6F 2E 70 6E 67 E6 98 AF E8 BE 93 E5 ; photo.png鏄  缁?
0001d8b0h: 85 A5 E6 BA 90 E6 96 87 E4 BB B6 EF BC 8C E8 AF ; 瑗婧愭枃浠讹紝璇
0001d8c0h: B7 E4 B8 8D E8 A6 81 E5 88 A0 E9 99 A4 E3 80 82 ; 蜂笃瑕佸埛閮ゃ€?
0001d8d0h: 0D 0A E4 BA 8C C2 B7 E3 80 81 E5 91 BD E4 BB A4 ; ..浜屽枫€佸懡浠?
0001d8e0h: E8 AF B4 E6 98 8E 0D 0A 2A 20 5B 75 73 61 67 65 ; 璇存槑..* [usage
0001d8f0h: 5D 3A 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ; ]:
0001d900h: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ;
```

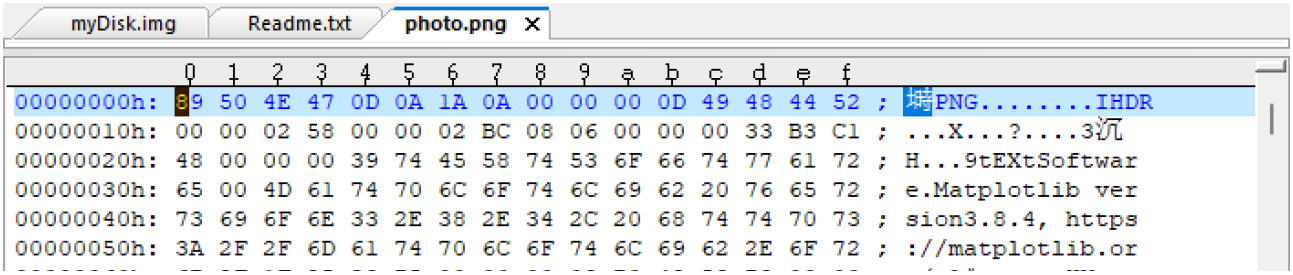
（2）将图片 photo.png 存入/home/photos 目录下。

```
[@ home ]$ cd photos

[@ photos ]$ fcreate photo
新建文件 photo 完成

[@ photos ]$ ls
当前目录: photos
子目录:
**********************************************
  文件类型      文件名      最后访问时间
**********************************************
  文件          photo       Sun May 19 00:59:25 2024


[@ photos ]$ fopen photo
打开文件 photo 成功，文件句柄 1

[@ photos ]$ fwrite 1 photo.png
当前文件指针：0，写入字节数：72065

[@ photos ]$ fsave

[@ photos ]$ _
```

查看文件卷，写入成功

```
            0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
000e1c0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000e1d0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000e1e0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000e1f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000e200h: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 ; 塒NG........IHDR
000e210h: 00 00 02 58 00 00 02 BC 08 06 00 00 00 33 B3 C1 ; ...X...?....3沉
000e220h: 48 00 00 00 39 74 45 58 74 53 6F 66 74 77 61 72 ; H...9tEXtSoftwar
000e230h: 65 00 4D 61 74 70 6C 6F 74 6C 69 62 20 76 65 72 ; e.Matplotlib ver
000e240h: 73 69 6F 6E 33 2E 38 2E 34 2C 20 68 74 74 70 73 ; sion3.8.4, https
000e250h: 3A 2F 2F 6D 61 74 70 6C 6F 74 6C 69 62 2E 6F 72 ; ://matplotlib.or
000e260h: 67 2F 1F 25 23 75 00 00 00 09 70 48 59 73 00 00 ; g/.%#u....pHYs..
000e270h: 0F 61 00 00 0F 61 01 A8 3F A7 69 00 01 00 00 49 ; .a...a.?  ....I
000e280h: 44 41 54 78 9C EC DD 77 5C 13 F7 FF 07 F0 57 D8 ; DATx滌鰡\.?.餡?
```

与 photo.png 文件一致



装 订 线

myDisk.img × | Readme.txt | photo.png

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0002ff30h: 18 B0 88 88 88 88 14 C6 80 45 44 44 44 A4 30 06 ; .蝝姿?苺EDDD?.
0002ff40h: 2C 22 22 22 22 85 31 60 11 11 11 11 29 8C 01 8B ; ,""""?`....)??
0002ff50h: 88 88 88 48 61 0C 58 44 44 44 44 0A 63 C0 22 22 ; 姿圍a.XDDDD.c?"
0002ff60h: 22 22 52 18 03 16 11 11 11 91 C2 FE 0F 2C 23 7A ; ""R......惇?,#z
0002ff70h: FF 25 00 B0 54 00 00 00 00 49 45 4E 44 AE 42 60 ;   %.癟....IEND葙`
0002ff80h: 82 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ?...............
0002ff90h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

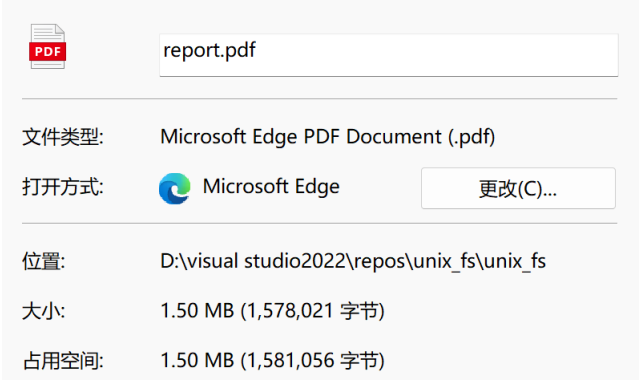
myDisk.img | Readme.txt | photo.png ×

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
000118f0h: 11 11 11 91 C2 18 B0 88 88 88 88 14 C6 80 45 44 ; ...惇.蝝姿?苺ED
00011900h: 44 44 A4 30 06 2C 22 22 22 22 85 31 60 11 11 11 ; DD?.,""""?`...
00011910h: 11 29 8C 01 8B 88 88 88 48 61 0C 58 44 44 44 44 ; .)?嬭姿Ha.XDDDD
00011920h: 0A 63 C0 22 22 22 22 52 18 03 16 11 11 11 91 C2 ; .c?""""R......惇
00011930h: 18 B0 88 88 88 88 14 C6 80 45 44 44 44 A4 30 06 ; .蝝姿?苺EDDD?.
00011940h: 2C 22 22 22 22 85 31 60 11 11 11 11 29 8C 01 8B ; ,""""?`....)??
00011950h: 88 88 88 48 61 0C 58 44 44 44 44 0A 63 C0 22 22 ; 姿圍a.XDDDD.c?"
00011960h: 22 22 52 18 03 16 11 11 11 91 C2 FE 0F 2C 23 7A ; ""R......惇?,#z
00011970h: FF 25 00 B0 54 00 00 00 00 49 45 4E 44 AE 42 60 ;   %.癟....IEND葙`
00011980h: 82                                             ; ?□
```

（2）将报告 report.pdf 存入/home/reports 目录下。

```
[@ home ]$ cd reports

[@ reports ]$ fcreate rep
新建文件 rep 完成

[@ reports ]$ ls
当前目录: reports
子目录:
**************************************************
  文件类型      文件名      最后访问时间
**************************************************
  文件          rep         Sun May 19 01:04:17 2024


[@ reports ]$ fopen rep
打开文件 rep 成功，文件句柄 2

[@ reports ]$ fwrite 2 report.pdf
当前文件指针：0，写入字节数：1578021

[@ reports ]$ fsave

[@ reports ]$
```

写入 1578021 个字节，很大，用到了二级索引。



与 report.pdf 内容一致

myDisk.img | Readme.txt | photo.png | report.pdf

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 25 50 44 46 2D 31 2E 37 0D 0A 25 B5 B5 B5 B5 0D ; %PDF-1.7..%档档.
00000010h: 0A 31 20 30 20 6F 62 6A 0D 0A 3C 3C 2F 54 79 70 ; .1 0 obj..<</Typ
00000020h: 65 2F 43 61 74 61 6C 6F 67 2F 50 61 67 65 73 20 ; e/Catalog/Pages
00000030h: 32 20 30 20 52 2F 4C 61 6E 67 28 7A 68 29 20 2F ; 2 0 R/Lang(zh) /
00000040h: 53 74 72 75 63 74 54 72 65 65 52 6F 6F 74 20 31 ; StructTreeRoot 1
00000050h: 30 30 20 30 20 52 2F 4D 61 72 6B 49 6E 66 6F 3C ; 00 0 R/MarkInfo<
00000060h: 3C 2F 4D 61 72 6B 65 64 20 74 72 75 65 3E 3E 2F ; </Marked true>>/
```

myDisk.img | Readme.txt | photo.png

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0002ffc0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
0002ffd0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
0002ffe0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
0002fff0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00030000h: 25 50 44 46 2D 31 2E 37 0D 0A 25 B5 B5 B5 B5 0D ; %PDF-1.7..%档档.
00030010h: 0A 31 20 30 20 6F 62 6A 0D 0A 3C 3C 2F 54 79 70 ; .1 0 obj..<</Typ
00030020h: 65 2F 43 61 74 61 6C 6F 67 2F 50 61 67 65 73 20 ; e/Catalog/Pages
00030030h: 32 20 30 20 52 2F 4C 61 6E 67 28 7A 68 29 20 2F ; 2 0 R/Lang(zh) /
00030040h: 53 74 72 75 63 74 54 72 65 65 52 6F 6F 74 20 31 ; StructTreeRoot 1
00030050h: 30 30 20 30 20 52 2F 4D 61 72 6B 49 6E 66 6F 3C ; 00 0 R/MarkInfo<
00030060h: 3C 2F 4D 61 72 6B 65 64 20 74 72 75 65 3E 3E 2F ; </Marked true>>/
00030070h: 4D 65 74 61 64 61 74 61 20 33 30 33 35 20 30 20 ; Metadata 3035 0
00030080h: 52 2F 56 69 65 77 65 72 50 72 65 66 65 72 65 6E ; R/ViewerPreferen
```

myDisk.img | Readme.txt | photo.png | report.pdf

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
001813c0h: 43 41 37 43 38 36 3E 3C 38 37 35 35 44 34 36 35 ; CA7C86><8755D465
001813d0h: 37 30 32 41 37 45 34 34 39 38 45 30 38 36 31 44 ; 702A7E4498E0861D
001813e0h: 35 39 43 41 37 43 38 36 3E 5D 20 2F 50 72 65 76 ; 59CA7C86>] /Prev
001813f0h: 20 31 35 31 36 39 31 31 2F 58 52 65 66 53 74 6D ;  1516911/XRefStm
00181400h: 20 31 35 31 30 38 38 32 3E 3E 0D 0A 73 74 61 72 ;  1510882>>..star
00181410h: 74 78 72 65 66 0D 0A 31 35 37 37 38 33 34 0D 0A ; txref..1577834..
00181420h: 25 25 45 4F 46                                  ; %%EOF
```

myDisk.img | Readme.txt | photo.png | report.pdf

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
001b47b0h: 32 41 37 45 34 34 39 38 45 30 38 36 31 44 35 39 ; 2A7E4498E0861D59
001b47c0h: 43 41 37 43 38 36 3E 3C 38 37 35 35 44 34 36 35 ; CA7C86><8755D465
001b47d0h: 37 30 32 41 37 45 34 34 39 38 45 30 38 36 31 44 ; 702A7E4498E0861D
001b47e0h: 35 39 43 41 37 43 38 36 3E 5D 20 2F 50 72 65 76 ; 59CA7C86>] /Prev
001b47f0h: 20 31 35 31 36 39 31 31 2F 58 52 65 66 53 74 6D ;  1516911/XRefStm
001b4800h: 20 31 35 31 30 38 38 32 3E 3E 0D 0A 73 74 61 72 ;  1510882>>..star
001b4810h: 74 78 72 65 66 0D 0A 31 35 37 37 38 33 34 0D 0A ; txref..1577834..
001b4820h: 25 25 45 4F 46 00 00 00 00 00 00 00 00 00 00 00 ; %%EOF...........
001b4830h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

装 订 线

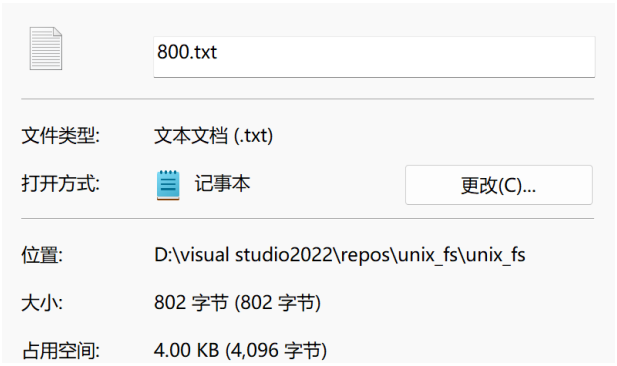
4.4 写入 800 字节文件测试

通过命令行方式测试：

- 新建文件/test/Jerry，打开该文件，任意写入 800 个字节；
- 将文件读写指针定位到第 500 字节，读出 500 个字节到字符串 abc。
- 将 abc 写回文件

（1）新建文件/test/Jerry，打开该文件，任意写入 800 个字节；
写入 800 字节文件

```
string content = "";
for (int i = 0; i < 800; i++) {
    string mid = to_string(i);
    content += mid;
    content += " ";
    if (content.length() >= 800) break;
}
cout << content.length() << " " << content << endl;
fd.write(content.c_str(), content.length());
```



运行程序

```
[@ root ]$ cd home

[@ home ]$ cd texts

[@ texts ]$ fcreate Jerry
新建文件 Jerry 完成

[@ texts ]$ ls
当前目录: texts
子目录:
************************************************
  文件类型     文件名     最后访问时间
************************************************
  文件         readme      Sun May 19 00:54:42 2024

  文件         Jerry       Sun May 19 01:15:50 2024


[@ texts ]$ fopen Jerry
打开文件 Jerry 成功，文件句柄 3

[@ texts ]$ fwrite 3 800.txt
当前文件指针：0，写入字节数：802

[@ texts ]$ _
```

查看文件卷，成功写入 800 字节

```
001b4a00h: 30 20 31 20 32 20 33 20 34 20 35 20 36 20 37 20 ; 0 1 2 3 4 5 6 7
001b4a10h: 38 20 39 20 31 30 20 31 31 20 31 32 20 31 33 20 ; 8 9 10 11 12 13
001b4a20h: 31 34 20 31 35 20 31 36 20 31 37 20 31 38 20 31 ; 14 15 16 17 18 1
001b4a30h: 39 20 32 30 20 32 31 20 32 32 20 32 33 20 32 34 ; 9 20 21 22 23 24
001b4a40h: 20 32 35 20 32 36 20 32 37 20 32 38 20 32 39 20 ;  25 26 27 28 29
001b4a50h: 33 30 20 33 31 20 33 32 20 33 33 20 33 34 20 33 ; 30 31 32 33 34 3
001b4a60h: 35 20 33 36 20 33 37 20 33 38 20 33 39 20 34 30 ; 5 36 37 38 39 40
001b4a70h: 20 34 31 20 34 32 20 34 33 20 34 34 20 34 35 20 ;  41 42 43 44 45
001b4a80h: 34 36 20 34 37 20 34 38 20 34 39 20 35 30 20 35 ; 46 47 48 49 50 5
```

（2）将文件读写指针定位到第 500 字节，读出 500 个字节

```
[@ texts ]$ fopen Jerry
打开文件 Jerry 成功，文件句柄 0

[@ texts ]$ fseek 0 500 beg
文件指针移动至 500 ,原本文件指针位于 802 ,移动距离 500

[@ texts ]$ fread 0 std 500
当前文件指针：500，文件大小：802，输出字节数：302
输出：
2 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 17
2 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 19
2 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 21
2 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227

[@ texts ]$
```

（3）将 abc 写回文件

```
[@ texts ]$ fseek 0 500 beg
文件指针移动至 500 ,原本文件指针位于 802 ,移动距离 500

[@ texts ]$ fread 0 b.txt 500
当前文件指针：500，文件大小：802，输出字节数：302
输出：成功输出到 b.txt 中！

[@ texts ]$
```

myDisk.img | Readme.txt | photo.png | report.pdf | b.txt

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 32 20 31 35 33 20 31 35 34 20 31 35 35 20 31 35 ; 2 153 154 155 15
00000010h: 36 20 31 35 37 20 31 35 38 20 31 35 39 20 31 36 ; 6 157 158 159 16
00000020h: 30 20 31 36 31 20 31 36 32 20 31 36 33 20 31 36 ; 0 161 162 163 16
00000030h: 34 20 31 36 35 20 31 36 36 20 31 36 37 20 31 36 ; 4 165 166 167 16
00000040h: 38 20 31 36 39 20 31 37 30 20 31 37 31 20 31 37 ; 8 169 170 171 17
00000050h: 32 20 31 37 33 20 31 37 34 20 31 37 35 20 31 37 ; 2 173 174 175 17
00000060h: 36 20 31 37 37 20 31 37 38 20 31 37 39 20 31 38 ; 6 177 178 179 18
00000070h: 30 20 31 38 31 20 31 38 32 20 31 38 33 20 31 38 ; 0 181 182 183 18
00000080h: 34 20 31 38 35 20 31 38 36 20 31 38 37 20 31 38 ; 4 185 186 187 18
00000090h: 38 20 31 38 39 20 31 39 30 20 31 39 31 20 31 39 ; 8 189 190 191 19
000000a0h: 32 20 31 39 33 20 31 39 34 20 31 39 35 20 31 39 ; 2 193 194 195 19
000000b0h: 36 20 31 39 37 20 31 39 38 20 31 39 39 20 32 30 ; 6 197 198 199 20
000000c0h: 30 20 32 30 31 20 32 30 32 20 32 30 33 20 32 30 ; 0 201 202 203 20
000000d0h: 34 20 32 30 35 20 32 30 36 20 32 30 37 20 32 30 ; 4 205 206 207 20
000000e0h: 38 20 32 30 39 20 32 31 30 20 32 31 31 20 32 31 ; 8 209 210 211 21
000000f0h: 32 20 32 31 33 20 32 31 34 20 32 31 35 20 32 31 ; 2 213 214 215 21
00000100h: 36 20 32 31 37 20 32 31 38 20 32 31 39 20 32 32 ; 6 217 218 219 22
00000110h: 30 20 32 32 31 20 32 32 32 20 32 32 33 20 32 32 ; 0 221 222 223 22
00000120h: 34 20 32 32 35 20 32 32 36 20 32 32 37 20       ; 4 225 226 227
```

所有操作完成，最终 img 大小

myDisk.img	2024/5/19 1:27	光盘映像文件	1,748 KB

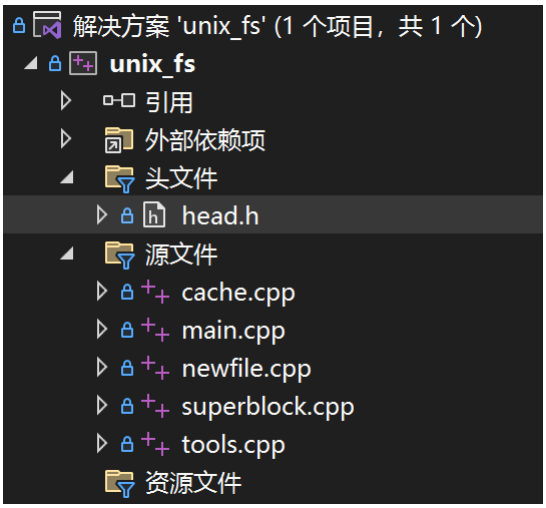
装

订

线

五，用户使用说明

项目结构：



运行方法：

在 visual studio 2022 环境下，开始执行此程序（CTRL + F5）。

所有命令可以通过 help 命令进行查看：

```
*************************************************************************************************
*                                                                                               *
*                                        Unix文件系统模拟                                       *
*                                                                                               *
* [usage]:                                                                                      *
* help                              [功能]:命令提示                                             *
* fformat                           [功能]:格式化文件系统                                       *
* ls                                [功能]:查看当前目录内容                                     *
* mkdir <dirname>                   [功能]:新建文件夹                                           *
* cd <dirname>                      [功能]:进入目录                                             *
* fcreate <filename>                [功能]:新建文件filename                                     *
* fopen <filename>                  [功能]:打开文件filename                                     *
* fwrite <fdnum> <infile> <size>    [功能]:从文件infile写入fdnum文件size字节                    *
* fwrite <fdnum> std                [功能]:从屏幕写入fdnum文件size字节                          *
* fread <fdnum> <outfile> <size>    [功能]:从fdnum文件读取size字节，输出到outfile              *
* fread <fdnum> std <size>          [功能]:从fdnum文件读取size字节，输出到屏幕                  *
* fseek <fdnum> <step> begin        [功能]:fdnum文件指针从开头偏移step                          *
* fseek <fdnum> <step> cur          [功能]:fdnum文件指针从现有位置偏移step                      *
* fseek <fdnum> <step> end          [功能]:fdnum文件指针从末尾偏移step                          *
* fflag <fdnum> read                [功能]:将fdnum文件权限更改为只读                           *
* fflag <fdnum> write               [功能]:将fdnum文件权限更改为可读写                         *
* fclose <fdnum>                    [功能]:关闭文件句柄为fdnum的文件                           *
* fdelete <filename>                [功能]:删除文件文件名为filename的文件或者文件夹            *
* exit                              [功能]:退出文件系统                                         *
* fsave                             [功能]:保存文件系统，cache写回（测试命令）                 *
* 注意：退出文件系统要使用exit命令！不能直接退出！                                             *
*************************************************************************************************
是否对文件系统进行初始化？[y/n]
```