Assignment Report: File System

120090584 Butian Xiong

1 Introduction [5']

1.1 xv6 file system

Let us first introduce the overall implementation method including 7 different layers.

- **File Descriptor** – Interface for user programs to access files.
- **Pathname** – Translates human-readable paths to file system entities.
- **Directory** – Manages lists of inodes, handling file creation.
- **Inode** – Stores metadata about files, excluding their name and content.
- **Logging** – Ensures atomic operations to maintain file system integrity.
- **Buffer Cache** – Caches disk blocks to minimize read/write operations.
- **Disk** – The physical storage medium for data.

Let us introduce the file system in the disk or device

- Boot Block: The first block on the disk, reserved for the boot loader.
- Superblock: Contains metadata about the file system, such as the size of the file system, the number of data blocks, the start of the inodes, and the start of the data blocks.
- Inode Table: A list of inodes that contain metadata about files. Each inode contains information like the type of the file (directory, file, device), its size, and pointers to data blocks.
- Bitmap (or free list): Tracks which blocks are free and which are in use.
- Data Blocks: The rest of the blocks on the disk, used to store file data or metadata (for directories).

Each inode in xv6 can have up to NDIRECT (12 in xv6) direct pointers, one singly indirect pointer, and one doubly indirect pointer. This structure determines the maximum file size.

There are also several drawbacks to this implementation:

- Maximum File Size: The file size in xv6 is limited by the number of block pointers an inode can hold. With direct, singly indirect, and doubly indirect pointers, the maximum file size can be limited.
- Symbolink: xv6 does not support symbolic links, which are a standard feature in modern Unix-like file systems for creating pointers to other files or directories.
- Large File: For very large files, accessing data blocks through indirect pointers can be inefficient due to the increased number of disk reads required.

1.2 Accomplishment

I have achieved the doubly linked storage mapping. The only problem I have met is that the log_write is sometimes not correct implemented. I think we need to first require our students to first understand the test process, and understand read() function's meaning. Then the progress will be much more smooth.

2 Implementation [3']

2.1 Bigfile

2.1.1 file.h

We need to first change the last line of inode structure. Since we change NDIRECT to 11

```
struct inode {
  uint dev; // Device number
  uint inum; // Inode number
  int ref; // Reference count
  struct sleeplock lock; // protects everything below here
  int valid; // inode has been read from disk?

  short type; // copy of disk inode
  short major;
  short minor;
  short nlink;
  uint size;
  uint addrs[NDIRECT+2];
};
```

2.1.2 fs.h

We also need to change the structure of dinode and some global variable

```
#define NDIRECT 11
#define NINDIRECT (BSIZE / sizeof(uint))
#define NDINDIRECT (NINDIRECT*NINDIRECT)
#define MAXFILE (NDIRECT + NINDIRECT + NDINDIRECT)

// On-disk inode structure
struct dinode {
  short type; // File type
  short major; // Major device number (T_DEVICE only)
  short minor; // Minor device number (T_DEVICE only)
  short nlink; // Number of links to inode in file system
  uint size; // Size of file (bytes)
  uint addrs[NDIRECT+2]; // Data block addresses
};
```

2.1.3 fs.c

We also need to implement bmap and itrunc. in bmap, we need to most of the things in if statement since we usually do not need to allocate a new disk space, we just need to return what is currently in used to the user. Here is my implementation of the third if statement, or say the doubly linked mapping:

```
    bn -= NINDIRECT;
  // if still larger, use doubly indirect link
  if (bn < NDINDIRECT){
    // Load indirect block, allocating if necessary.
    // doubly linked indirect location is at NDIRECT+1
    if((addr = ip->addrs[NDIRECT+1]) == 0){
      // allocate a block of address point to a block of address (first
          red↪ layer)
      addr = balloc(ip->dev);
      if(addr == 0) // no space return 0
```

```
      return 0;
    //create the indirect block
    ip->addrs[NDIRECT+1] = addr;
  }
  // read that particular disk space we just allocate or in the memory
  bp = bread(ip->dev, addr);
  // get the begining of that block as an address for the first layer
  a = (uint*)bp->data;
  // if that block has not been used
  if((addr = a[bn/NINDIRECT]) == 0){
    // then we allocate a new address
    addr = balloc(ip->dev);
    //
    if(addr){
      // if allocate successful
      // we write to that particular buffer a+bn as the new block address
          red↪ we assign
      a[bn/NINDIRECT] = addr;
      // commit to bp
      log_write(bp);
    }
  }
  // release the block
  brelse(bp);

  bs = bread(ip->dev, addr);
  // get the begining of that block as an address for the first layer
  b = (uint*)bs->data;
  // if that block has not been used
  if ((addr = b[bn%NINDIRECT])==0){
    // then we allocate a new address
    addr = balloc(ip->dev);
    //
    if(addr){
      // if allocate successful
      // we write to that particular buffer a+bn as the new block address
          red↪ we assign
      b[bn%NINDIRECT] = addr;
      brelse(bs);
      log_write(bs);
      return addr;
      // commit to bp
    }
    else{
      brelse(bs);
      return 0;
    }
    // release the block
  }
  brelse(bs);
  return addr;
}
```

As for the itrunc, it would be much easier:

```
  if(ip->addrs[NDIRECT+1]){
  bp = bread(ip->dev, ip->addrs[NDIRECT]);
  a = (uint*)bp->data;
  for(j = 0; j < NINDIRECT; j++){
```

```
    if(a[j]){
      bs = bread(ip->dev, a[j]);
      b = (uint*)bs->data;
      for (k=0; k<NINDIRECT; k++){
        if(b[k]){
          bfree(ip->dev, b[k]);
        }
      }
      brelse(bs);
      bfree(ip->dev, a[j]);
    }
  }
```

2.2 Symlink

3 Test [2']

- first, we need to balloc twice in the third if statement in bmap, this will bas the write blocks test
- put the log_write in the correct location help us solve pass the read test

Briefly discuss which part of your implementation helped you pass each test. (Several sentences for each subsection should suffice.)

3.1 symlink (if done)

4 Comments [0']

I think would it be possible to follow the MIT structure from the first assignment. Since the first assignment does not make any sense to me. And I think it would be great not to use latex to type a report. Although latex is how we usually use it during conferences or journals, but even if the assignment instruction is written in notion, why expect students to write latex code.

But the last two assignment is much greater than the first and second, thanks for your efforts.