日曜研究室

技術的な観点から日常を綴ります

[xv6 #60] Chapter 5 – File system – Inodes

テキストの69〜70ページ

本文

inodeという用語は関連する2つの意味を持つ。
ひとつは、ファイルのサイズとデータブロックのセクタ番号のリストを含むディスク上のデータ構造を指す。
もうひとつは、ディスク上のinodeのコピーはもちろん、カーネルで必要となる追加の情報を含む、メモリ上のデータ構造を指す。

すべてのディスク上のinodeは、inodeブロックと呼ばれるディスクの連続したエリアの中にまとめられる。
それぞれのinodeは同じサイズなので、nという番号を与え、ディスク上のn番目のinodeを捜すのは簡単である。
実際、この番号nは、inode番号もしくはi-numberと呼ばれ、実装中でinodeを特定する方法となる。

ディスク上のinodeは、dinode構造体として定義される。
typeというフィールドは、ファイルか、ディレクトリか、特殊なファイル（デバイス）かを区別するためにある。
typeがゼロだったら、そのディスク上のinodeは空きであるということを意味する。

fs.h

```
01 // On-disk file system format.
02 // Both the kernel and user programs use this header file.
03
04 // Block 0 is unused.  Block 1 is super block.
05 // Inodes start at block 2.
06
07 #define ROOTINO 1  // root i-number
08 #define BSIZE 512  // block size
09
10 // File system super block
11 struct superblock {
```

```
12    uint size;          // Size of file system image (blocks)
13    uint nblocks;       // Number of data blocks
14    uint ninodes;       // Number of inodes.
15    uint nlog;          // Number of log blocks
16  };
17
18  #define NDIRECT 12
19  #define NINDIRECT (BSIZE / sizeof(uint))
20  #define MAXFILE (NDIRECT + NINDIRECT)
21
22  // On-disk inode structure
23  struct dinode {
24    short type;           // File type
25    short major;          // Major device number (T_DEV only)
26    short minor;          // Minor device number (T_DEV only)
27    short nlink;          // Number of links to inode in file system
28    uint size;            // Size of file (bytes)
29    uint addrs[NDIRECT+1];   // Data block addresses
30  };
31
32  // Inodes per block.
33  #define IPB           (BSIZE / sizeof(struct dinode))
34
35  // Block containing inode i
36  #define IBLOCK(i)     ((i) / IPB + 2)
37
38  // Bitmap bits per block
39  #define BPB           (BSIZE*8)
40
41  // Block containing bit for block b
42  #define BBLOCK(b, ninodes) (b/BPB + (ninodes)/IPB + 3)
43
44  // Directory is a file containing a sequence of dirent structures.
45  #define DIRSIZ 14
46
47  struct dirent {
48    ushort inum;
49    char name[DIRSIZ];
50  };
```

カーネルは、よく使われるinodeのまとまりをメモリ上に保持する。
inode構造体は、ディスク上のdinode構造体のメモリ上のコピーとして使われる。
カーネルは、Cポインタがあるinodeを参照してるときだけ、そのinodeをメモリに格納する。
そして、参照カウントがゼロになったら、メモリからそのinodeを破棄する。
iget関数、iput関数は、ひとつのinodeへのポインタを獲得したり解放したりしつつその参照カウントを変更する。
あるinodeに対するポインタは、ファイルディスクリプタや、現在の作業ディレクトリや、execのような一時的なカーネルのコードによって生成される。

file.h

```
01  struct file {
02    enum { FD_NONE, FD_PIPE, FD_INODE } type;
03    int ref; // reference count
04    char readable;
05    char writable;
06    struct pipe *pipe;
07    struct inode *ip;
```

```
08   uint off;
09 };
10
11
12 // in-core file system types
13
14 struct inode {
15   uint dev;           // Device number
16   uint inum;          // Inode number
17   int ref;            // Reference count
18   int flags;          // I_BUSY, I_VALID
19
20   short type;         // copy of disk inode
21   short major;
22   short minor;
23   short nlink;
24   uint size;
25   uint addrs[NDIRECT+1];
26 };
27
28 #define I_BUSY 0x1
29 #define I_VALID 0x2
30
31 // device implementations
32
33 struct devsw {
34   int (*read)(struct inode*, char*, int);
35   int (*write)(struct inode*, char*, int);
36 };
37
38 extern struct devsw devsw[];
39
40 #define CONSOLE 1
```

iget関数が返すinode構造体は、有用な内容を何も含んでいないだろう。
ディスク上のinodeのコピーを保持することを保証するため、ilock関数を呼ばなければならない。
この関数は、inodeをロックし（他のプロセスが同じinodeに対してilock出来ないようにするために）、そして読み取り済みでなければ、ディスクからinodeを読み取る。
iunlock関数は、inodeに対するロックを解放する。
inodeポインタの獲得を、ロックから分離することは、いくつかの場合にデッドロックを避ける助けになる。
例えば、ディレクトリを探索する場合である。
複数のプロセスは、iget関数によって返されたあるinodeへのCポインタを保持することが出来るが、一度に一つのプロセスだけがそのinodeをロック出来る。

inodeキャッシュは、カーネルのコードやCポインタを保持するデータ構造のために、inodeをキャッシュするのみである。
inodeキャッシュの主な役割は、複数のプロセスによるアクセスを確実に同期化することであり、キャッシュすることではない。
あるinodeが頻繁に利用される場合、inodeキャッシュによってキャッシュされなくても、バッファキャッシュによって、メモリ上に適切に保持される。

感想

inodeの概要です。
ヘッダのソースは載せてますが、各関数のソースは後の節で説明があるはずなので、今回は載せてません。

メモリ上にinodeの複製を持つのは、主にキャッシュの為ではなく、同じinodeに対するアクセスを同期化する目的の為のようです。

カテゴリー: 技術 | タグ: xv6 | 投稿日: 2012/4/4 水曜日 [http://peta.okechan.net/blog/archives/1627] |
