Lab 实验报告 IX

Lab: File System

丁睿

dromniscience@gmail.com

更新：2020 年 12 月 19 日

目录

1 任务完成清单

Subtask	Done?	Time
Large Files	Y	1h
Symbolic Links	Y	2h

Grade: 100/100

我的作业已开放在这个网页上。

2 详细情况 & 困难和收获

本次 lab 并未深入文件系统的深入细节。只需注意 RISCV xv6 文件系统层次中的加锁约定以及几个灵活好用的系统例程的使用即可顺利完成作业。代码中加入了详细的注释。部分细节不在此处展开。

2.1 Large Files

我们需要修改普通文件的 inode 表示，采用二级混合索引。这和类 Unix 操作系统的设计思路是一致的。我们使用以下两幅图说明差别。注意到这样的拓展很容易延伸到三级索引甚至更多级的索引。在 xv6 中，考虑到硬盘的实际存储为 200,000 个块，二级索引足够了。

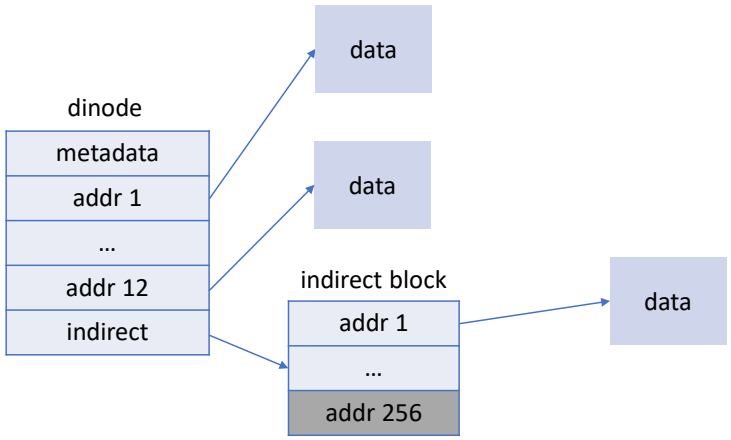

图 1: 原本普通文件的 inode 节点设计，除元数据区外前 12 个块迎接映射物理块号，第 13 块采用一级索引。每个条目占`sizeof(uint)`，即 4 字节。每块大小为`BSIZE`，即 1KB。灰色部分表示该条目暂未使用，为无效。

由于文件系统采用七层分层设计，而文件在 inode 层中才得以实现，且其提供给目录层以及更上层的接口均被封装，因此我们只需修改 inode 层中的有关数据结构和函数。我们这里列出需要修改的函数如下:

`bmap` 给定 inode 号和逻辑磁盘块号，翻译成物理地址块号。

itrunc 截断文件。用于删除文件或者以O_TRUNC参数打开可读文件。

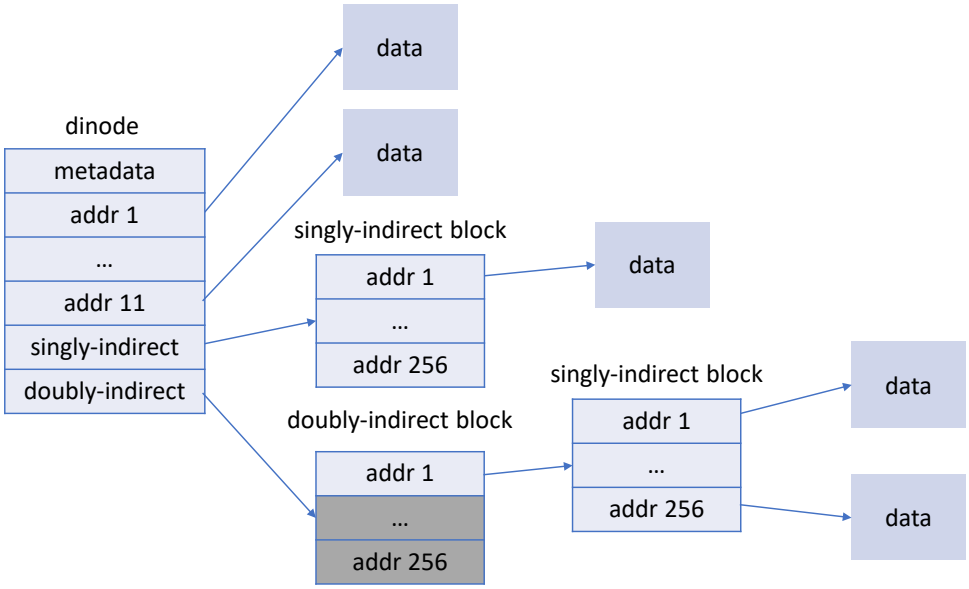

图 2: 使用二级索引的 inode 节点设计，除元数据区外前 11 个块迎接映射物理块号，第 12 块采用一级索引，第 13 块采用二级索引。每个条目占sizeof(uint)，即 4 字节。每块大小为BSIZE，即 1KB。灰色部分表示该条目暂未使用，为无效。注意到由于dinode的大小不可改变，因此必须牺牲一个直接索引条目的空间存放二级索引，故直接索引只有 11 个。

由于源码中已有一级索引的实现，只需模仿实现二级索引即可。具体的细节不做展开。注意以下两点即可：

- bread后必须及时brelse以消除引用，释放资源 (如果 buffer cache 的引用计数归零)。
- bmap和itrunc都需要在一个 transaction 当中被调用 (粗略地说，它们被begin_op和end_op包围)。由于itrunc并不需要修改索引表页和数据页的内容，因此不必为它们打 log。唯一需要打 log 的是对dinode的数据域的修改，但是因为iupdate函数封装了log_write，故可以很简单地完成工作。当然，在bmap中，由于可能更新多级索引表的内容，必须直接调用log_write打log。

2.2 Symbolic Links

我们需要支持软链接——即建立一种特殊的文件，它引用其他文件。而本身只需要记录被引对象的位置。不同于硬链接必须通过访问当前文件系统的 inode 号来找到相应文件，它可以实现跨文件系统的文件对象引用。

在我的设计中，为了尽可能使用原来的文件读写函数，我没有特别考虑储存的效率和安全性——一个软连接文件引用文件的路径直接按照普通文件的方式存在它的数据块中。这样它的 inode 结构完全一致，只是由于该路径被限制为至多MAXPATH(128)，因此混合索引表的后 12 项不会被使用。为了简单，我总是在其中存放一个长为 128 的字符数组，以消除长度检查，如图3所示。

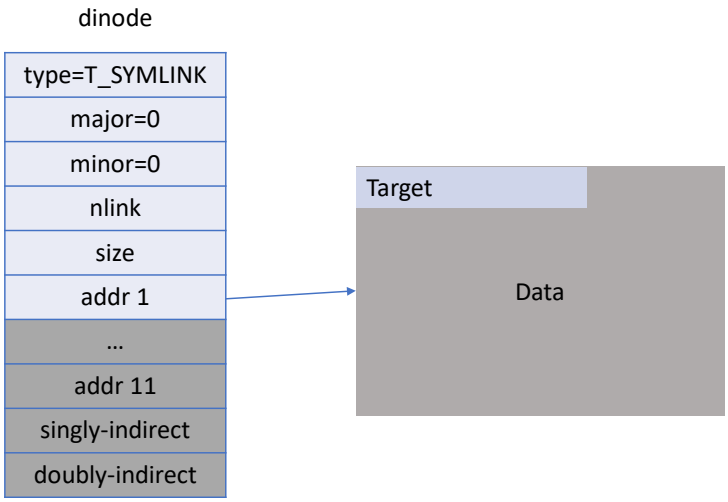

图 3: 软连接文件的存储方式。灰色部分表示分配了但实际永不会不会使用的空间。

通过`symlink`创建软连接文件时，由于要写硬盘数据，必须将修改封装在 transaction 中，即应当调用`begin_op`预分配一个 log 的空间，调用`log_write`写 log，最后`end_op`向磁盘写入 log，打上 commit 标记，并开始真正地将脏页写回磁盘。其余注意的地方就是每个系统级函数调用时加锁和引用的情况。这些内容均已详细地注释在 kernel/sysfile.c:`sys_symlink`和`sys_open`中，此处不再赘述。为了对`sys_open`制作最小的修改，我们只需要在取得`ip`后和分配文件表表目前根据`ip->type`是否为`T_SYMLINK`以及`omode`中有无声明`NOFOLLOW`选择追溯软链接对象。一旦超过追溯步数超过`DEPTH`，则认为发生了循环软链接 (cyclic soft links)。注意，每轮在取得新的`inode`表项前，要调用`iunlockput`以释放锁，并且减少引用计数。锁和引用是在`namex`函数里完成的。如未如此操作，将导致死锁。

本文最后摘录了`sys_open`中的这段新加的代码，我觉得这可能是最简单的修改方式了。它插入的位置上面已经说明。

3 参考资料

1. *xv6: a simple, Unix-like teaching operating system*
   Russ Cox, Frans Kaashoek, Robert Morris *August 31, 2020*
2. *The RISC-V Reader: An Open Architecture Atlas*
   David Patterson, Andrew Waterman *1st Edition*
3. 现代操作系统 [M]
   A.S.Tanenbaum, H. Bos 著, 陈向群 马洪兵 译, 北京: 机械工业出版社, 2011: 47-95

```c
/* kernel/sysfile.c:sys_open() */
  // Fs : symlink
  // find the correct inode if the file is a soft link
  // note that ip->lock is held
  if(ip->type == T_SYMLINK && !(omode & O_NOFOLLOW)){
    // search at depth 10
    int depth = 0;
    char path[MAXPATH];
    while(depth < DEPTH && ip->type == T_SYMLINK){
      readi(ip, 0, (uint64)path, 0, MAXPATH);
      iunlockput(ip); // release ip->lock and decrement the
          refcnt in time

      if((ip = namei(path)) == 0){
        end_op();
        return -1;
      }
      ilock(ip);
      depth++;
    }

    // cyclic links
    if(depth == DEPTH){
      iunlockput(ip); // release ip->lock and decrement the
          refcnt in time
      end_op();
      return -1;
    }
  }
```

对sys_open所做的修改