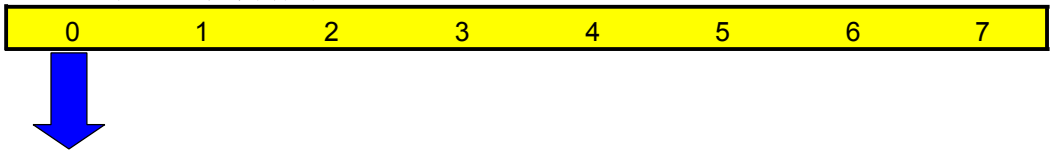
下面列出主要的文件系统数据结构，便于查询，主要参见 include/linux/fs.h

一、超级块（超辑块是与设备一一对应的）

超级块数组（在内存中）



{

节点数
逻辑块数
i 节点所占用的数据块数
逻辑块位图所占用的数据块数
第一个数据逻辑块号
$\log_2$(数据块数/逻辑块)
文件最大长度
文件系统幻数

用于磁盘与内存

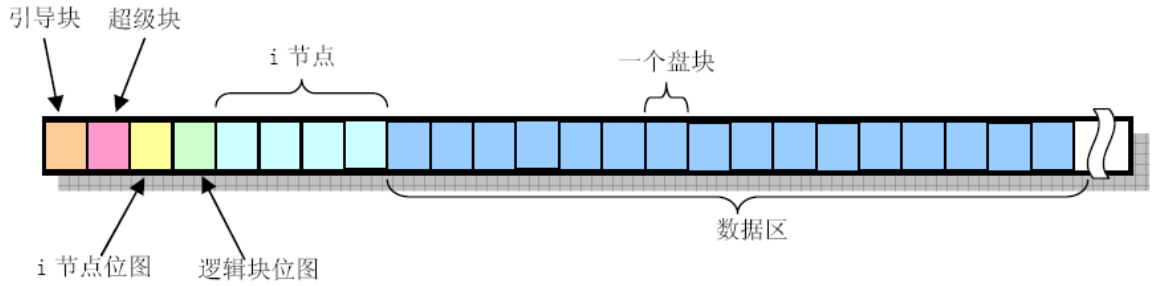
---

i 节点位图缓冲区指针数组
逻辑块位图缓冲区指针数组
超级块所在的设备号
被安装的文件系统根目录的 i 节点
被安装到的 i 节点
修改时间
等待该超级块的进程
被锁定标志
只读标志
已修改（脏）标志

仅用于内存

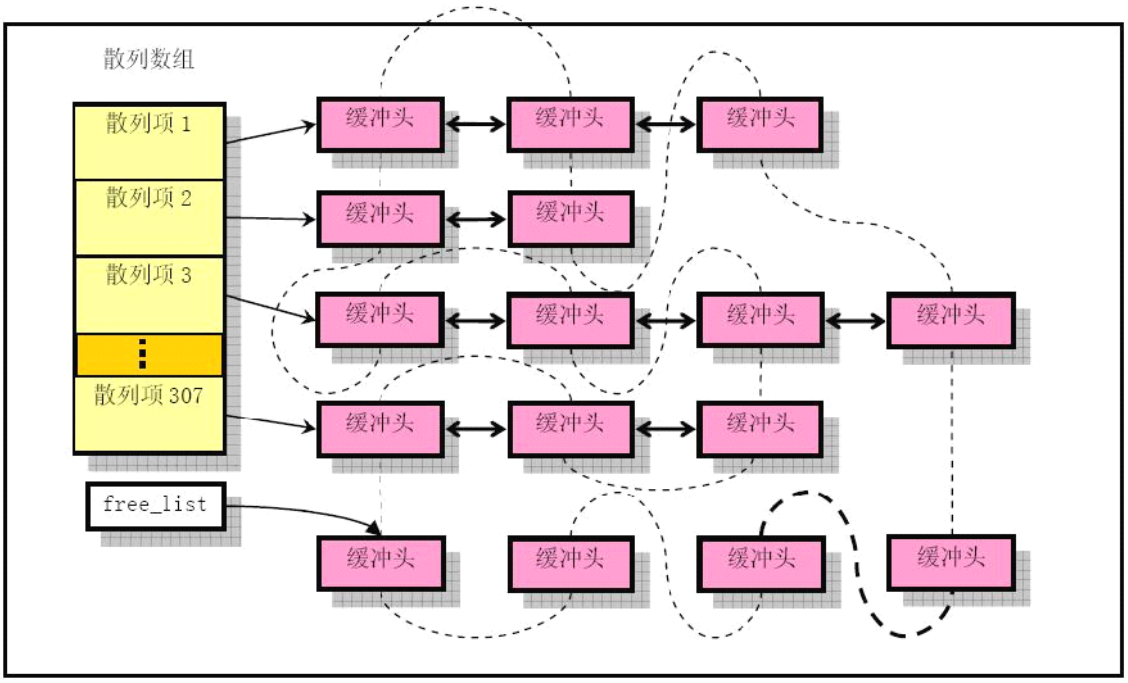
}

超级块（在磁盘上）



二、缓冲区散列队列（缓冲区是与盘块一一对应的）

下面给出赵博画的缓冲区的散列队列逻辑示意图



以上散列项，free_list 与缓冲头均为 buffer_head 结构

{

指向 1024 个字节数据块的指针

块号

数据源的设备号

更新标志：表示数据是否已更新

修改（脏）标志

使用的用户数

缓冲区是否被锁定

等待该缓冲区解锁的任务

hash 队列上前一块

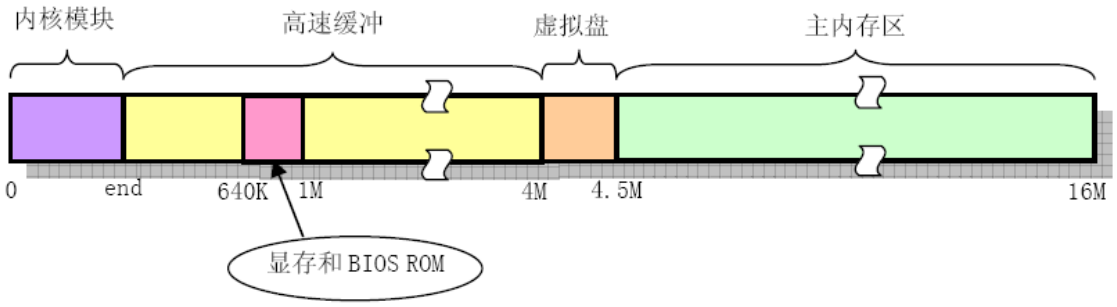
hash 队列上后一块

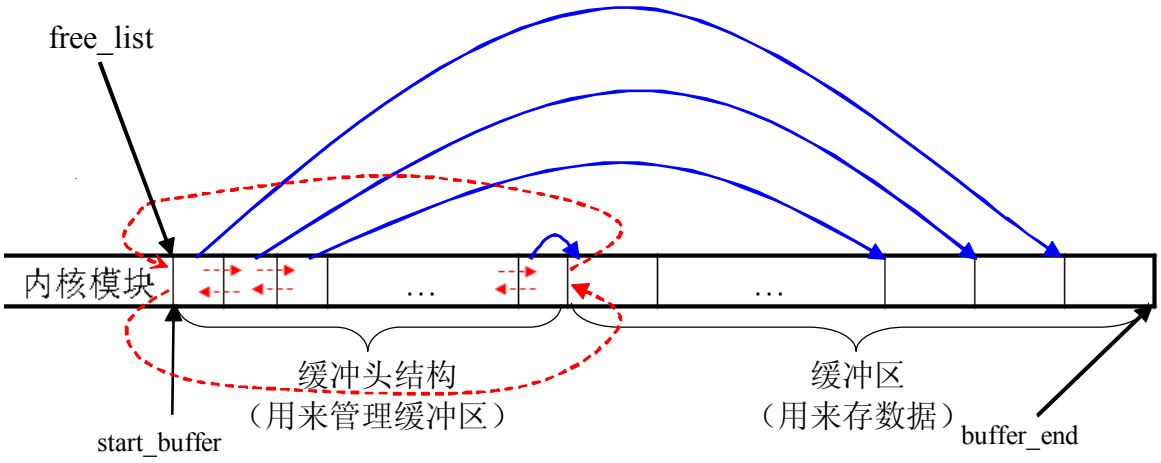
空闲表上前一块

空闲表上后一块

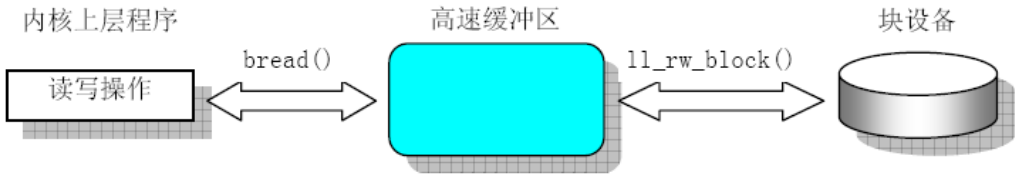
}

在 Linux0.11 内核中具体实现时，在物理内存中缓冲区的状态如下：





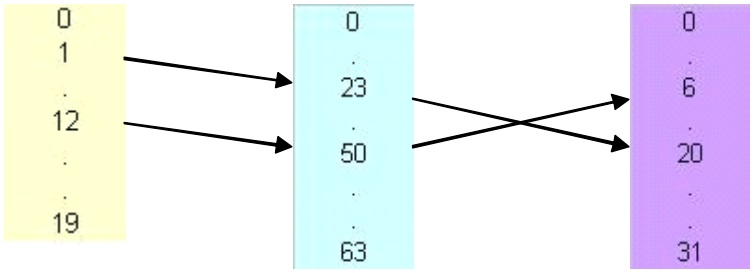
以上实线箭头为 data 指针，一个缓冲头指向一个大小为 1KB 的缓冲区，虚线是空闲链表指针，构成双向链表，初始化时是相邻的缓冲头链接成空闲表，位于内核模块的 struct buffer_head *型数组经哈希函数计算后指向缓冲头结构，初始化时为全 NULL。哈希函数=（设备号 异或 块号） 取余 307，307 为哈希表长。hash_table 在实际运行后的图参见前面的缓冲区的散列队列逻辑示意图。



缓冲区与块设备（如磁盘）间的读写操作均由 ll_rw_block()实现，在 kernel\blk_drv\ll_rw_blk.c 中。缓冲区是文件系统的底层，介于读写磁盘的驱动程序与文件系统之间，它对文件系统提供了磁盘盘块的抽象，对上接口为设备号和块号。

三、索引节点（索引节点是与文件一一对应的）

用户文件描述符表　　文件表　　（内存）索引节点表



用户文件描述符表与进程有关，即 task_struct 中的 filp 数组，每个进程只能有 20 个打开文件。索引节点表中的索引节点与文件一一对应，即整个系统在某一时刻只能打开 32 个文件。文件表用来关联这两张表，它的主要功用是记录文件的当前读写指针的偏移值。

文件表结构：

{

　　文件操作模式（RW 位）

　　文件打开和控制的标志

　　对应文件句柄（文件描述符）数

　　指向对应（内存）i 节点指针

　　文件位置（读写偏移值）

}

索引节点结构：

{

文件类型和属性（rwx 位）

用户 id（文件拥有者标识符）

文件大小（字节数）

修改时间

组 id（文件拥有者所在的组）

文件目录项链接数

直接（0~6），间接（7）或双重间接（8）逻辑块号

} 既在磁盘又在内存中

等待该 i 节点的进程

最后访问时间

i 节点自身修改时间

i 节点所在的设备号

i 节点号

i 节点被使用的次数

锁定标志

已修改（脏）标志

管道标志

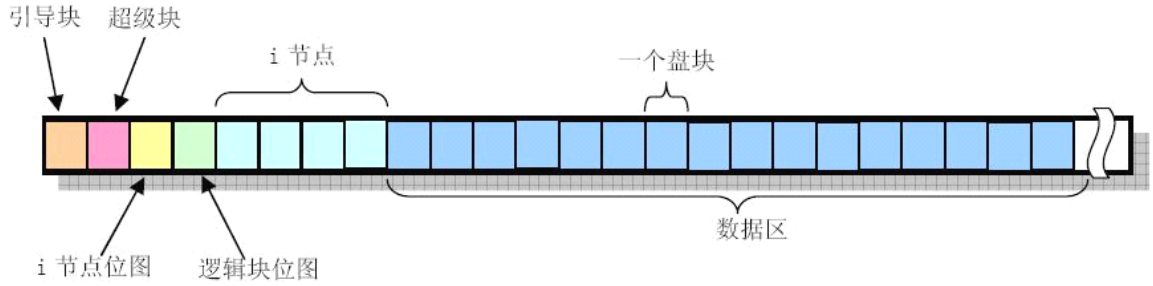
安装标志

搜索（lseek）标志

更新标志

} 仅在内存中

索引节点有可能在内存和不在内存。若在内存表明该 i 节点当前活跃，可在 i 节点表（见 p257 inode.c 的 15 行）找到。若不在内存，须从设备中读取，方法是根据该 i 节点所在设备找到该 i 节点的超级块，根据超级块中的信息，利用以下公式可计算出 i 节点所在的逻辑块号。



上图为赵博画的在块设备中文件系统各部分的布局。

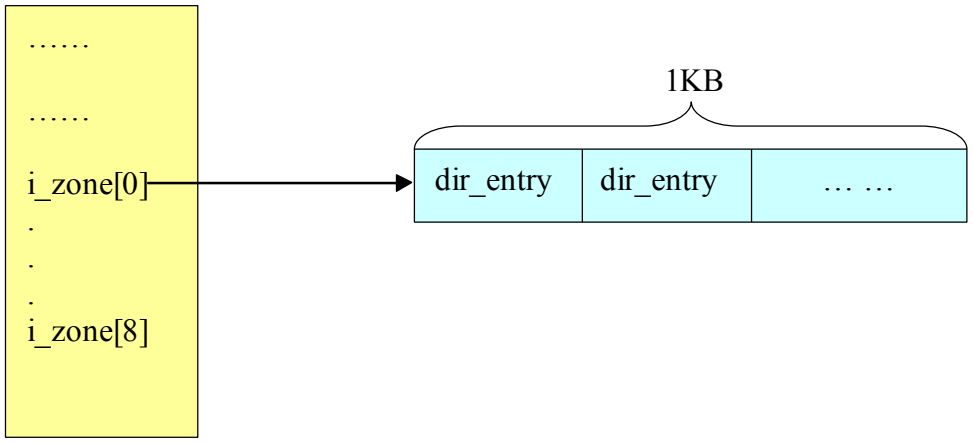
所以 i 节点逻辑块号=2+i 节点位图在内存中对应的超级块所占块数+（i 节点号-1）/每块含有的 i 节点数

（引导块+超级块）

上图中 i 节点位图和逻辑块位图在内存中对应为超级块指向的缓冲区，i 节点存放在 i 节点表中，数据区通常被读到缓冲区，由缓冲区管理。

无论是文件还是目录的 i 节点，都将文件实际数据块记录在 i_zone 中。目录文件对应的目录项结构与普通文件一样存储在 i_zone 指明的逻辑块中。操作目录文件的函数有 find_entry, add_entry, sys_mkdir 等。

目录文件的 i 节点：



对于读写文件操作来说，系统给出了两种实现方式，直接根据设备号与读写指针（存在文件表中）读块设备。根据 i 节点与文件表指针读文件逻辑块。前者是读块设备上连续的块，后者是读文件意义上连续的块，要通过 bmap 转换成块设备意义上的块。算法上两个实现只有细微差别。

关于管道的书上 p297 图很能说明问题。