程序 12-3 linux/fs/truncate.c

```
1  /*
2   *  linux/fs/truncate.c
3   *
4   *  (C) 1991  Linus Torvalds
5   */
6
7  #include <linux/sched.h>  // 调度程序头文件，定义了任务结构 task_struct、任务 0 数据等。
8
9  #include <sys/stat.h>     // 文件状态头文件。含有文件或文件系统状态结构 stat{}和常量。
10
   //// 释放所有一次间接块。（内部函数）
   // 参数 dev 是文件系统所在设备的设备号；block 是逻辑块号。成功则返回 1，否则返回 0。
11 static int free_ind(int dev,int block)
12 {
13         struct buffer_head * bh;
14         unsigned short * p;
15         int i;
16         int block_busy;                              // 有逻辑块没有被释放的标志。
17
   // 首先判断参数的有效性。如果逻辑块号为 0，则返回。然后读取一次间接块，并释放其上表
   // 明使用的所有逻辑块，然后释放该一次间接块的缓冲块。 函数 free_block()用于释放设备
   // 上指定逻辑块号的磁盘块（fs/bitmap.c 第 47 行）。
18         if (!block)
19                 return 1;
20         block_busy = 0;
21         if (bh=bread(dev,block)) {
22                 p = (unsigned short *) bh->b_data;  // 指向缓冲块数据区。
23                 for (i=0;i<512;i++,p++)             // 每个逻辑块上可有 512 个块号。
24                         if (*p)
25                                 if (free_block(dev,*p)) {  // 释放指定的设备逻辑块。
26                                         *p = 0;            // 清零。
27                                         bh->b_dirt = 1;    // 设置已修改标志。
28                                 } else
29                                         block_busy = 1;    // 设置逻辑块没有释放标志。
30                 brelse(bh);                                // 然后释放间接块占用的缓冲块。
31         }
   // 最后释放设备上的一次间接块。但如果其中有逻辑块没有被释放，则返回 0（失败）。
32         if (block_busy)
33                 return 0;
34         else
35                 return free_block(dev,block);      // 成功则返回 1，否则返回 0。
36 }
37
   //// 释放所有二次间接块。
   // 参数 dev 是文件系统所在设备的设备号；block 是逻辑块号。
38 static int free_dind(int dev,int block)
39 {
40         struct buffer_head * bh;
41         unsigned short * p;
42         int i;
43         int block_busy;                              // 有逻辑块没有被释放的标志。
44
```

```
    // 首先判断参数的有效性。如果逻辑块号为 0，则返回。然后读取二次间接块的一级块，并释
    // 放其上表明使用的所有逻辑块，然后释放该一级块的缓冲块。
45          if (!block)
46                  return 1;
47          block_busy = 0;
48          if (bh=bread(dev,block)) {
49                  p = (unsigned short *) bh->b_data;  // 指向缓冲块数据区。
50                  for (i=0;i<512;i++,p++)             // 每个逻辑块上可连接 512 个二级块。
51                          if (*p)
52                                  if (free_ind(dev,*p)) {   // 释放所有一次间接块。
53                                          *p = 0;           // 清零。
54                                          bh->b_dirt = 1;   // 设置已修改标志。
55                                  } else
56                                          block_busy = 1;   // 设置逻辑块没有释放标志。
57                  brelse(bh);                         // 释放二次间接块占用的缓冲块。
58          }
    // 最后释放设备上的二次间接块。但如果其中有逻辑块没有被释放，则返回 0（失败）。
59          if (block_busy)
60                  return 0;
61          else
62                  return free_block(dev,block);
63  }
64
    //// 截断文件数据函数。
    // 将节点对应的文件长度截为 0，并释放占用的设备空间。
65  void truncate(struct m_inode * inode)
66  {
67          int i;
68          int block_busy;                             // 有逻辑块没有被释放的标志。
69
    // 首先判断指定 i 节点有效性。如果不是常规文件、目录文件或链接项，则返回。
70          if (!(S_ISREG(inode->i_mode) || S_ISDIR(inode->i_mode) ||
71               S_ISLNK(inode->i_mode)))
72                  return;
    // 然后释放 i 节点的 7 个直接逻辑块，并将这 7 个逻辑块项全置零。函数 free_block()用于
    // 释放设备上指定逻辑块号的磁盘块（fs/bitmap.c 第 47 行）。若有逻辑块忙而没有被释放
    // 则置块忙标志 block_busy。
73  repeat:
74          block_busy = 0;
75          for (i=0;i<7;i++)
76                  if (inode->i_zone[i]) {             // 如果块号不为 0，则释放之。
77                          if (free_block(inode->i_dev,inode->i_zone[i]))
78                                  inode->i_zone[i]=0;   // 块指针置 0。
79                          else
80                                  block_busy = 1;       // 若没有释放掉则置标志。
81                  }
82          if (free_ind(inode->i_dev,inode->i_zone[7]))  // 释放所有一次间接块。
83                  inode->i_zone[7] = 0;                 // 块指针置 0。
84          else
85                  block_busy = 1;                       // 若没有释放掉则置标志。
86          if (free_dind(inode->i_dev,inode->i_zone[8])) // 释放所有二次间接块。
87                  inode->i_zone[8] = 0;                 // 块指针置 0。
88          else
```

```
89                  block_busy = 1;                         // 若没有释放掉则置标志。
    // 此后设置 i 节点已修改标志，并且如果还有逻辑块由于“忙”而没有被释放，则把当前进程
    // 运行时间片置 0，以让当前进程先被切换去运行其他进程，稍等一会再重新执行释放操作。
90          inode->i_dirt = 1;
91          if (block_busy) {
92                  current->counter = 0;                   // 当前进程时间片置 0。
93                  schedule();
94                  goto repeat;
95          }
96          inode->i_size = 0;                              // 文件大小置零。
    // 最后重新置文件修改时间和 i 节点改变时间为当前时间。宏 CURRENT_TIME 定义在头文件
    // include/linux/sched.h 第 142 行处，定义为(startup_time + jiffies/HZ)。用于取得从
    // 1970:0:0:0 开始到现在为止经过的秒数。
97          inode->i_mtime = inode->i_ctime = CURRENT_TIME;
98  }
99
100
```