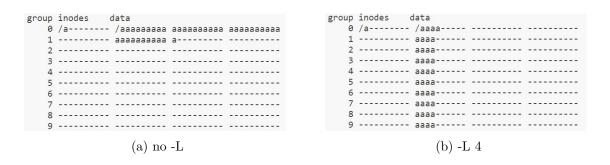
## HW12

姓名: 陈锐林, 学号:21307130148 2023 年 12 月 14 日

# Chapter41-FFS

### Question1:

由题意可以知道, in.largefile 里只有一行命令"file /a 40"; 这意味着该文件是放在 root 下,即第一行开始的。如果我们不使用-L 标志,数据块就会从第一行开始堆叠排放,如下左图;而如果使用-L 4,每个 gruop 都占 4 个坑,如下右图。



Question2:-L 30 和 -L 4 的意义是一样的;即每行分配 30 个位置。但是因为文件 a 只有 40 的大小,因此排到第二行多 11 个就停止了,如下图。

Question4: 首先分析 in.manyfiles 内的文件,有三种;一是直接归属于 root 的,还有分别属于子目录 j 和 t 的。那么显然每一类的文件应该在同一组内,而且能发现,这里文件大小都不大;一组的数据块完全放得下。所以最后结果如下图所述,不同目录下的文件放在同一 group。

## Chapter 42-Journal

### Question1:

(1) 首先观察通过指令 fsck.py -D 生成的如下文件系统。inode 中的"[f a:-1]" 表示有俩个文件为空。而观察中的首项 (根目录),排除与后面有连接的 g 和 w,m 和 z 应该是两个文件名; inode 中的"[d a:0]" 是根目录,"[d a:12]" 和"[d a:6]" 是两个目录项,分别对应 data 中的数据块,再根据前面剩下 w 和 g,可以得到目录名;最后注意到 g 下还有"(s,15)",可知 g/s 也是一个文件。汇总结果,directory: "/","/g","/w"; file: "/m","/z","/g/s"。

```
Final state of file system:

inode bitmap 1000100010001010
inodes        [d a:0 r:4] [] [] [] [d a:12 r:2] [] [] [] [d a:6 r:2] [] [] [] [f a:-1 r:2] [] [f a:-1 r:1]
data bitmap 100000100001000
data        [(.,0) (..,0) (g,8) (w,4) (m,13) (z,13)] [] [] [] [] [(.,8) (..,0) (s,15)] [] [] [] [] [] (.,4) (..,0) [] [] []
```

(2) 对于其他随机种子,-s1生成如下;类似的可得结果。directory: "/","/a","/m" (先找到与后面的数据块有连接的 a 和 m, 如"(m,7)" 和"(.,7)" 匹配); file: "/g", "/m/m","/m/e","/a/r","/a/w"。(接着找到目录下的文件)

(3) 对随机种子,-s 2 生成如下;结果也可得。directory: "/","/c","/c/o","/c/o,"/c/o/u"。 (现在根目录找到下一级目录 c,接着再在 c 的项中找到 o,以此类推); file: "/c/o/q", "/c/o/u/q","/c/o/u/e"。

```
inode bitmap 1000000100110101
inodes        [d a:0 r:3] [] [] [] [] [] [d a:4 r:3] [] [] [f a:-1 r:1] [f a:-1 r:2] [] [d a:11 r:3] []
data bitmap 100010000010001
data        [(.,0) (..,0) (c,13)] [] [] [] (.,7) (..,13) (u,15) (q,11)] [] [] [] [] [] [] (.,13) (..,0)
        (o,7)] [] [] [] [(.,15) (..,7) (q,11) (e,10)]
```

(4) 对随机种子,-s 3 生成如下;结果也可得。directory: "/","/r","/r/s"; file: "/f","/x"。

### Question2:

(1) 运行指令 fsck.py -S 1 后,能得到如下结果。能发现问题出在 inode bitmap 上,按照题目中的 [f a:-1 r:2] 来看,inode bitmap 的第十三位不应该是 0,该是 1。 (2) 修改位图,以改变其标识,让它存在。

## Chapter43-LFS

### Question1:

(1) 能看到下面出现了 ku3 和 qg9, 所以应该进行 create file 这俩个文件。中间有个区域 (如下图) 正好对应剩下的一句命令,由 imap 和 size(ptrs) 的变化可知, 应该是写了 ku3, offset 为 7, size 为 4。总结如下: create file /ku3 write file /ku3 offset=7 size=4 create file /qg9 (2) 就三条指令,顺序是显然的。(3) 最后活跃的块应该包括: 临界区, qg9 相关, ku3 写相关的。即 0,8,9,11,12,13,14。(4) 当将-n 调整为 5 后, 很明显发现工作量变大, 这时共 23 各块, 且有新的文件引入,并且操作类型应该也是变得更复杂的; 不单单只是 create 和 write。(如下图)