第四十章-文件系统实现

预备知识

使用工具 vsfs.py 来研究文件系统状态如何随着各种操作的发生而改变。文件系统以空状态开始，只有一个根目录。模拟发生时，会执行各种操作，从而慢慢改变文件系统的磁盘状态。详情请参阅 README 文件。

可能进行的操作有：

- mkdir() - 创建文件夹
- creat() - 创建新的空文件
- open(), write(), close() - 将一个块添加到文件
- link() - 创建一个文件的硬链接
- unlink() - 删除一个硬链接 (如果 linkcnt==0,删除文件)

要了解此作业的功能，您必须首先了解如何表示此文件系统的磁盘状态。
通过打印四种不同数据结构的内容来显示文件系统的状态：

inode bitmap - 标识哪些 inode 已经被分配
inodes - inode 表及内容
data bitmap - 标识哪些数据块已经被分配
data - 数据块的内容

位图应该非常容易理解，其中 1 表示分配了相应的 inode 或数据块，0 表示该 inode 或数据块是空闲的。

每个 inodes 都有三个字段：第一个字段指示文件的类型（例如，f 表示常规文件，d 表示目录）；
第二个指示哪个数据块属于该文件（在这里，数据块的地址为 -1 表示文件是空的，地址为非负数时,表示文件引用的数据块地址）；
第三个显示文件或目录的引用计数。 例如，以下索引节点是一个常规文件，该文件为空（地址字段设置为 -1 ），并且在文件系统中只有一个链接：

```
[f a:-1 r:1]
```

如果为该文件分配了一个块（例如块 10），则将显示如下：

```
[f a:10 r:1]
```

如果有人随后创建了到该文件的硬链接，它将变成：

```
[f a:10 r:2]
```

最后，数据块可以保留用户数据或目录数据。如果保存的是目录数据，则块中的每个条目的格式均为（name，inumber），其中 “name” 是文件或目录的名称，“inumber” 是文件的 inode 号。
因此，假设根的 inode 为 0，则空的根目录如下所示：

```
[(.,0) (..,0)]
```

如果我们在根目录中添加一个已分配索引号 1 的文件 “f”，则根目录内容将变为：

```
[(.,0) (..,0) (f,1)]
```

如果数据块保存的是用户数据，则该数据块仅显示为单个字符，例如 “h”。 如果为空且未分配，则仅显示一对空括号（[]）。

因此，整个文件系统如下所示：

```
inode bitmap 11110000
inodes       [d a:0 r:6] [f a:1 r:1] [f a:-1 r:1] [d a:2 r:2] [] ...
data bitmap  11100000
data         [(.,0) (..,0) (y,1) (z,2) (f,3)] [u] [(.,3) (..,0)] [] ...
```

该文件系统具有八个 inode 和八个数据块。 根目录包含三个条目（不包括".","..")："y"，"z"和 "f"。通过查找 inode 1，我们可以看到 "y" 是一个常规文件（类型为 f），并为其分配了一个数据块（地址 1）。数据块 1 中是文件 "y"的内容：即 "u"。 我们还可以看到 "z" 是一个空的常规文件（地址字段设置为-1），而 "f"（ inode 编号 3 ）是一个目录，也为空。 您还可以从位图中看到前四个 inode 位图条目被标记为已分配，以及前三个数据位图条目被标记为已分配。

可以使用以下标志运行模拟器：

```
prompt>python2 vsfs.py -h
Usage: vsfs.py [options]

Options:
  -h, --help            显示此帮助消息并退出
  -s SEED, --seed=SEED  指定随机种子
  -i NUMINODES, --numInodes=NUMINODES
                        文件系统中的 inode 数量
  -d NUMDATA, --numData=NUMDATA
                        文件系统中数据块的数量
  -n NUMREQUESTS, --numRequests=NUMREQUESTS
                        要求模拟的操作数量
  -r, --reverse         不打印状态,而是打印操作
  -p, --printFinal      打印所有文件/目录
  -c, --compute         计算结果
```

典型的用法是简单地指定一个随机种子（以产生一个不同的问题）以及模拟请求的数量。
在此默认模式下，模拟器会在每个步骤中打印出文件系统的状态，并询问您必须执行哪种操作才能将文件系统从一种状态转换为另一种状态。 例如：

```
python2 vsfs.py -n 6 -s 16
ARG seed 16
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
```

```
inodes         [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap    10000000
data           [(.,0) (..,0) (y,1)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap   11000000
inodes         [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap    11000000
data           [(.,0) (..,0) (y,1)] [u] [] [] [] [] [] []

Which operation took place?

inode bitmap   11000000
inodes         [d a:0 r:2] [f a:1 r:2] [] [] [] [] [] []
data bitmap    11000000
data           [(.,0) (..,0) (y,1) (m,1)] [u] [] [] [] [] [] []

Which operation took place?

inode bitmap   11000000
inodes         [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap    11000000
data           [(.,0) (..,0) (y,1)] [u] [] [] [] [] [] []

Which operation took place?

inode bitmap   11100000
inodes         [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap    11000000
data           [(.,0) (..,0) (y,1) (z,2)] [u] [] [] [] [] [] []

Which operation took place?

inode bitmap   11110000
inodes         [d a:0 r:3] [f a:1 r:1] [f a:-1 r:1] [d a:2 r:2] [] [] [] []
data bitmap    11100000
data           [(.,0) (..,0) (y,1) (z,2) (f,3)] [u] [(.,3) (..,0)] [] [] [] [] []
```

在这种模式下运行时，模拟器仅显示一系列状态，并询问是什么操作导致了这些状态的变化。 使用 "-c" 标志运行会向我们显示答案。

```
python2 vsfs.py -n 6 -s 16 -c
ARG seed 16
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap   10000000
inodes         [d a:0 r:2] [] [] [] [] [] [] []
data bitmap    10000000
data           [(.,0) (..,0)] [] [] [] [] [] [] []
```

```
creat("/y");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1)] [] [] [] [] [] [] []

fd=open("/y", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1)] [u] [] [] [] [] [] []

link("/y", "/m");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1) (m,1)] [u] [] [] [] [] [] []

unlink("/m");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1)] [u] [] [] [] [] [] []

creat("/z");

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1) (z,2)] [u] [] [] [] [] [] []

mkdir("/f");

inode bitmap  11110000
inodes        [d a:0 r:3] [f a:1 r:1] [f a:-1 r:1] [d a:2 r:2] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (y,1) (z,2) (f,3)] [u] [(.,3) (..,0)] [] [] [] [] []
```

具体来说，创建了文件 "/y"，在其上附加了一个块，创建了从 "/m" 到 "/y" 的硬链接，通过调用 unlink 删除了 "/m"，创建了文件 "/z"，并创建目录 "/f"。

您还可以在“反向”模式（带有 "-r" 标志）下运行模拟器，打印操作而不是状态，以查看是否可以预测给定操作的状态变化：

```
python2 vsfs.py -n 6 -s 16 -r
ARG seed 16
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse True
ARG printFinal False

Initial state
```

```
inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

creat("/y");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/y", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

link("/y", "/m");

  State of file system (inode bitmap, inodes, data bitmap, data)?

unlink("/m");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/z");

  State of file system (inode bitmap, inodes, data bitmap, data)?

mkdir("/f");

  State of file system (inode bitmap, inodes, data bitmap, data)?
```

其他一些标志控制着模拟器的各个方面，包括 inodes 的数量（"-i"），数据块的数量（"-d"）以及是否打印文件系统中所有目录和文件的最终列表（"-p"）。

Problem1

问题描述

用一些不同的随机种子（比如 17、18、19、20）运行模拟器，看看你是否能确定每次状态变化之间一定发生了哪些操作。

问题分析

首先弄清不同操作会使磁盘状态发生何种改变：

- mkdir() - 创建文件夹：修改inode位图，增加一个inode用来存放新目录元数据，向存放新目录的目录块中增加一个条目，修改data位图，增加一个数据块用于存放新目录的内容，更新相应inode中的引用计数
- creat() - 创建新的空文件：修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数
- open(), write(), close() - 将一个块添加到文件：修改data位图，增加一个数据块用于存放文件的新内容，修改inode中的数据块地址字段
- link() - 创建一个文件的硬链接：修改inode，增加其中的引用计数，在保存链接的目录块中增加一个条目
- unlink() - 删除一个硬链接 (如果 linkcnt==0,删除文件)：修改inode，减小其中的引用计数，在保存链接的目录块中删除一个条目，当引用计数减为0时，删除文件，释放inode、数据块，修改inode位图、data位图

然后根据磁盘状态的改变确定发生了何种操作。

问题解答

seed 17

```
python2 vsfs.py -n 6 -s 17
ARG seed 17
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (u,1)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (u,1) (a,2)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (u,1)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:4] [d a:1 r:2] [d a:2 r:2] [] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (u,1) (z,2)] [(.,1) (..,0)] [(.,2) (..,0)] [] [] []
[] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:5] [d a:1 r:2] [d a:2 r:2] [d a:3 r:2] [] [] [] []
data bitmap   11110000
data          [(.,0) (..,0) (u,1) (z,2) (s,3)] [(.,1) (..,0)] [(.,2) (..,0)]
[(.,3) (..,0)] [] [] [] []
```

```
Which operation took place?

inode bitmap  11111000
inodes        [d a:0 r:5] [d a:1 r:2] [d a:2 r:2] [d a:3 r:2] [f a:-1 r:1] [] []
[]
data bitmap   11110000
data          [(.,0) (..,0) (u,1) (z,2) (s,3)] [(.,1) (..,0)] [(.,2) (..,0)
(x,4)] [(.,3) (..,0)] [] [] [] []
```

操作1同时修改了inode位图和data位图，只有mkdir()可以做到。查看1号inode,发现新建了一个目录，其数据存放在1号数据块，在0号数据块中查看新增加的条目，指示新建的目录名为“u”，所以操作1是mkdir("/u")

操作2只修改了inode位图，所以是creat()。查看2号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“a”，所以操作2是creat("/a")

操作3修改了inode位图，删除了inode和目录块中的条目，所以是unlink()。发现删除的是2号inode，所以操作3是unlink("/a")

操作4同时修改了inode位图和data位图，只有mkdir()可以做到。查看2号inode,发现新建了一个目录，其数据存放在2号数据块，在0号数据块中查看新增加的条目，指示新建的目录名为“z”，所以操作4是mkdir("/z")

操作5同时修改了inode位图和data位图，只有mkdir()可以做到。查看3号inode,发现新建了一个目录，其数据存放在3号数据块，在0号数据块中查看新增加的条目，指示新建的目录名为“s”，所以操作5是mkdir("/s")

操作6只修改了inode位图，所以是creat()。查看4号inode,发现新建了一个文件，在3号数据块（目录z的目录块）中查看新增加的条目，指示新建的文件名为“x”，所以操作6是creat("/z/x")

seed 18

```
python2 vsfs.py -n 6 -s 18
ARG seed 18
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (f,1)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
```

```
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (f,1) (s,2)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:4] [d a:1 r:2] [f a:-1 r:1] [d a:2 r:2] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (f,1) (s,2) (h,3)] [(.,1) (..,0)] [(.,3) (..,0)] []
[] [] [] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:4] [d a:1 r:2] [f a:3 r:1] [d a:2 r:2] [] [] [] []
data bitmap   11110000
data          [(.,0) (..,0) (f,1) (s,2) (h,3)] [(.,1) (..,0)] [(.,3) (..,0)] [f]
[] [] [] []

Which operation took place?

inode bitmap  11111000
inodes        [d a:0 r:4] [d a:1 r:2] [f a:3 r:1] [d a:2 r:2] [f a:-1 r:1] [] []
[]
data bitmap   11110000
data          [(.,0) (..,0) (f,1) (s,2) (h,3)] [(.,1) (..,0) (o,4)] [(.,3)
(..,0)] [f] [] [] [] []

Which operation took place?

inode bitmap  11111100
inodes        [d a:0 r:4] [d a:1 r:2] [f a:3 r:1] [d a:2 r:2] [f a:-1 r:1] [f
a:-1 r:1] [] []
data bitmap   11110000
data          [(.,0) (..,0) (f,1) (s,2) (h,3) (c,5)] [(.,1) (..,0) (o,4)] [(.,3)
(..,0)] [f] [] [] [] []
```

操作1同时修改了inode位图和data位图，只有mkdir()可以做到。查看1号inode,发现新建了一个目录，其数据存放在1号数据块，在0号数据块中查看新增加的条目，指示新建的目录名为“f”，所以操作1是mkdir("/f")

操作2只修改了inode位图，所以是creat()。查看2号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“s”，所以操作2是creat("/s")

操作3同时修改了inode位图和data位图，只有mkdir()可以做到。查看3号inode,发现新建了一个目录，其数据存放在2号数据块，在0号数据块中查看新增加的条目，指示新建的目录名为“h”，所以操作3是mkdir("/h")

操作4修改了data位图，修改了2号inode（文件s）中的地址字段，增加了3号数据块，所以操作4是fd=open("/s", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd)

操作5只修改了inode位图，所以是creat()。查看4号inode,发现新建了一个文件，在1号数据块（目录f的目录块）中查看新增加的条目，指示新建的文件名为“o”，所以操作5是creat("/f/o")

操作6只修改了inode位图，所以是creat()。查看5号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“c”，所以操作6是creat("/c")

seed 19

```
python2 vsfs.py -n 6 -s 19
ARG seed 19
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (k,1)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (k,1) (g,2)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (k,1) (g,2)] [g] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (k,1) (g,2) (b,1)] [g] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:3] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (k,1) (g,2) (b,1) (t,1)] [g] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
```

```
data          [(.,0) (..,0) (g,2) (b,1) (t,1)] [g] [] [] [] [] []
```

操作1只修改了inode位图，所以是creat()。查看1号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“k”，所以操作1是creat("/k")

操作2只修改了inode位图，所以是creat()。查看2号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“g”，所以操作2是creat("/g")

操作3修改了data位图，修改了1号inode（文件k）中的地址字段，增加了1号数据块，所以操作3是fd=open("/k", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd)

操作4增加了1号inode（文件k）中的引用计数字段，所以是link()。在0号数据块中查看新增加的条目，指示新建的链接名为“b”，所以操作4是link("/k","/b")

操作5增加了1号inode（文件k）中的引用计数字段，所以是link()。在0号数据块中查看新增加的条目，指示新建的链接名为“t”，所以操作5是link("/k","/t")

操作6减小了1号inode（文件k）中的引用计数字段，所以是unlink()。在0号数据块中查看删除的条目，指示删除的链接名为“k”，所以操作6是unlink("/k")

seed 20

```
python2 vsfs.py -n 6 -s 20
ARG seed 20
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (x,1)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (x,1)] [x] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (x,1) (k,2)] [x] [] [] [] [] [] []
```

```
Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (x,1) (k,2) (y,3)] [x] [] [] [] [] []

Which operation took place?

inode bitmap  10110000
inodes        [d a:0 r:2] [] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (k,2) (y,3)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  10100000
inodes        [d a:0 r:2] [] [f a:-1 r:1] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (k,2)] [] [] [] [] [] [] []
```

操作1只修改了inode位图，所以是creat()。查看1号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“x”，所以操作1是creat("/x")

操作2修改了data位图，修改了1号inode（文件x）中的地址字段，增加了1号数据块，所以操作2是fd=open("/x", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd)

操作3只修改了inode位图，所以是creat()。查看2号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“k”，所以操作3是creat("/k")

操作4只修改了inode位图，所以是creat()。查看3号inode,发现新建了一个文件，在0号数据块中查看新增加的条目，指示新建的文件名为“y”，所以操作4是creat("/y")

操作5修改了inode位图和data位图，删除了inode、数据块和目录块中的条目，所以是unlink()。发现删除的是1号inode（文件x），所以操作5是unlink("/x")

操作6修改了inode位图，删除了inode和目录块中的条目，所以是unlink()。发现删除的是3号inode（文件y），所以操作6是unlink("/y")

答案验证

seed 17

```
python2 vsfs.py -n 6 -s 17 -c
ARG seed 17
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []
```

```
mkdir("/u");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (u,1)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/a");

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (u,1) (a,2)] [(.,1) (..,0)] [] [] [] [] [] []

unlink("/a");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (u,1)] [(.,1) (..,0)] [] [] [] [] [] []

mkdir("/z");

inode bitmap  11100000
inodes        [d a:0 r:4] [d a:1 r:2] [d a:2 r:2] [] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (u,1) (z,2)] [(.,1) (..,0)] [(.,2) (..,0)] [] [] []
[] []

mkdir("/s");

inode bitmap  11110000
inodes        [d a:0 r:5] [d a:1 r:2] [d a:2 r:2] [d a:3 r:2] [] [] [] []
data bitmap   11110000
data          [(.,0) (..,0) (u,1) (z,2) (s,3)] [(.,1) (..,0)] [(.,2) (..,0)]
[(.,3) (..,0)] [] [] [] []

creat("/z/x");

inode bitmap  11111000
inodes        [d a:0 r:5] [d a:1 r:2] [d a:2 r:2] [d a:3 r:2] [f a:-1 r:1] [] []
[]
data bitmap   11110000
data          [(.,0) (..,0) (u,1) (z,2) (s,3)] [(.,1) (..,0)] [(.,2) (..,0)
(x,4)] [(.,3) (..,0)] [] [] [] []
```

seed 18

```
python2 vsfs.py -n 6 -s 18 -c
ARG seed 18
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state
```

```
inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/f");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (f,1)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/s");

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (f,1) (s,2)] [(.,1) (..,0)] [] [] [] [] [] []

mkdir("/h");

inode bitmap  11110000
inodes        [d a:0 r:4] [d a:1 r:2] [f a:-1 r:1] [d a:2 r:2] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (f,1) (s,2) (h,3)] [(.,1) (..,0)] [(.,3) (..,0)] []
[] [] [] []

fd=open("/s", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11110000
inodes        [d a:0 r:4] [d a:1 r:2] [f a:3 r:1] [d a:2 r:2] [] [] [] []
data bitmap   11110000
data          [(.,0) (..,0) (f,1) (s,2) (h,3)] [(.,1) (..,0)] [(.,3) (..,0)] [f]
[] [] [] []

creat("/f/o");

inode bitmap  11111000
inodes        [d a:0 r:4] [d a:1 r:2] [f a:3 r:1] [d a:2 r:2] [f a:-1 r:1] [] []
[]
data bitmap   11110000
data          [(.,0) (..,0) (f,1) (s,2) (h,3)] [(.,1) (..,0) (o,4)] [(.,3)
(..,0)] [f] [] [] [] []

creat("/c");

inode bitmap  11111100
inodes        [d a:0 r:4] [d a:1 r:2] [f a:3 r:1] [d a:2 r:2] [f a:-1 r:1] [f
a:-1 r:1] [] []
data bitmap   11110000
data          [(.,0) (..,0) (f,1) (s,2) (h,3) (c,5)] [(.,1) (..,0) (o,4)] [(.,3)
(..,0)] [f] [] [] [] []
```

seed 19

```
python2 vsfs.py -n 6 -s 19 -c
ARG seed 19
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

creat("/k");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (k,1)] [] [] [] [] [] [] []

creat("/g");

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (k,1) (g,2)] [] [] [] [] [] [] []

fd=open("/k", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (k,1) (g,2)] [g] [] [] [] [] [] []

link("/k", "/b");

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (k,1) (g,2) (b,1)] [g] [] [] [] [] [] []

link("/b", "/t");

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:3] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (k,1) (g,2) (b,1) (t,1)] [g] [] [] [] [] [] []

unlink("/k");

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
```

```
data          [(.,0) (..,0) (g,2) (b,1) (t,1)] [g] [] [] [] [] []
```

seed 20

```
python2 vsfs.py -n 6 -s 20 -c
ARG seed 20
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

creat("/x");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (x,1)] [] [] [] [] [] [] []

fd=open("/x", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (x,1)] [x] [] [] [] [] [] []

creat("/k");

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (x,1) (k,2)] [x] [] [] [] [] [] []

creat("/y");

inode bitmap  11110000
inodes        [d a:0 r:2] [f a:1 r:1] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (x,1) (k,2) (y,3)] [x] [] [] [] [] [] []

unlink("/x");

inode bitmap  10110000
inodes        [d a:0 r:2] [] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (k,2) (y,3)] [] [] [] [] [] [] []

unlink("/y");

inode bitmap  10100000
```

```
inodes          [d a:0 r:2] [] [f a:-1 r:1] [] [] [] [] []
data bitmap     10000000
data            [(.,0) (..,0) (k,2)] [] [] [] [] [] [] []
```

经验证，答案正确。

Problem2

问题描述

现在使用不同的随机种子（比如 21、22、23、24），但使用 -r 标志运行，这样做可以让你在显示操作时猜测状态的变化。关于 inode 和数据块分配算法，根据它们喜欢分配的块，你可以得出什么结论?

问题分析

首先弄清不同操作会使磁盘状态发生何种改变：

- mkdir() - 创建文件夹：修改inode位图，增加一个inode用来存放新目录元数据，向存放新目录的目录块中增加一个条目，修改data位图，增加一个数据块用于存放新目录的内容，更新相应inode中的引用计数
- creat() - 创建新的空文件：修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数
- open(), write(), close() - 将一个块添加到文件：修改data位图，增加一个数据块用于存放文件的新内容，修改inode中的数据块地址字段
- link() - 创建一个文件的硬链接：修改inode，增加其中的引用计数，在保存链接的目录块中增加一个条目
- unlink() - 删除一个硬链接 (如果 linkcnt==0,删除文件)：修改inode，减小其中的引用计数，在保存链接的目录块中删除一个条目，当引用计数减为0时，删除文件，释放inode、数据块，修改inode位图、data位图

然后根据操作更新磁盘状态。

问题解答

seed 21

```
python2 vsfs.py -n 6 -s 21 -r
ARG seed 21
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/o");

  State of file system (inode bitmap, inodes, data bitmap, data)?
```

```
creat("/b");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/o/q");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/b", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/o/q", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/o/j");

  State of file system (inode bitmap, inodes, data bitmap, data)?
```

操作1是mkdir("/o")，修改inode位图，增加一个inode用来存放新目录元数据，向存放新目录的目录块中增加一个条目，修改data位图，增加一个数据块用于存放新目录的内容，更新相应inode中的引用计数，所以状态1为：

```
inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (o,1)] [(.,1) (..,0)] [] [] [] [] [] []
```

操作2是creat("/b")，修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数，所以状态2为：

```
inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0)] [] [] [] [] [] []
```

操作3是creat("/o/q")，修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数，所以状态3为：

```
inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3)] [] [] [] [] [] []
```

操作4是fd=open("/b", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd)，修改data位图，增加一个数据块用于存放文件的新内容，修改inode中的数据块地址字段，所以状态4为：

```
inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:2 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3)] [m] [] [] [] [] []
```

操作5是fd=open("/o/q", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd)，修改data位图，增加一个数据块用于存放文件的新内容，修改inode中的数据块地址字段，所以状态5为：

```
inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:2 r:1] [f a:3 r:1] [] [] [] []
data bitmap   11110000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3)] [m] [j] [] [] [] []
```

操作6是creat("/o/j")，修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数，所以状态6为：

```
inode bitmap  11111000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:2 r:1] [f a:3 r:1] [f a:-1 r:1] [] []
[]
data bitmap   11110000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3) (j,4)] [m] [j] [] []
[] []
```

seed 22

```
python2 vsfs.py -n 6 -s 22 -r
ARG seed 22
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

creat("/z");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

unlink("/z");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/y");

  State of file system (inode bitmap, inodes, data bitmap, data)?

link("/y", "/s");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/e");
```

```
  State of file system (inode bitmap, inodes, data bitmap, data)?
```

操作1是creat("/z")，修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数，所以状态1为：

```
inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (z,1)] [] [] [] [] [] [] []
```

操作2是fd=open("/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd)，修改data位图，增加一个数据块用于存放文件的新内容，修改inode中的数据块地址字段，所以状态2为：

```
inode bitmap  11000000
inodes        [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (z,1)] [q] [] [] [] [] [] []
```

操作3是unlink("/z")，修改inode，减小其中的引用计数，在保存链接的目录块中删除一个条目，当引用计数减为0时，删除文件，释放inode、数据块，修改inode位图、data位图，删除目录块中的相应条目，所以状态3为：

```
inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []
```

操作4是creat("/y")，修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数，所以状态4为：

```
inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1)] [] [] [] [] [] [] []
```

操作5是link("/y", "/s")，修改inode，增加其中的引用计数，在保存链接的目录块中增加一个条目，所以状态5为：

```
inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:2] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1) (s,1)] [] [] [] [] [] [] []
```

操作6是creat("/e")，修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数，所以状态6为：

```
inode bitmap  11100000
inodes        [d a:0 r:2] [f a:-1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1) (s,1) (e,2)] [] [] [] [] [] [] []
```

针对其余随机数种子的做法都是相同 的，为节约篇幅，在此直接使用-c标志查看答案。

seed 23

```
python2 vsfs.py -n 6 -s 23 -r -c
ARG seed 23
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/c");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (c,1)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/c/t");

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (c,1)] [(.,1) (..,0) (t,2)] [] [] [] [] [] []

unlink("/c/t");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (c,1)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/c/q");

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (c,1)] [(.,1) (..,0) (q,2)] [] [] [] [] [] []

creat("/c/j");

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (c,1)] [(.,1) (..,0) (q,2) (j,3)] [] [] [] [] [] []

link("/c/q", "/c/h");

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:2] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
```

```
data          [(.,0) (..,0) (c,1)] [(.,1) (..,0) (q,2) (j,3) (h,2)] [] [] [] []
[] []
```

seed 24

```
python2 vsfs.py -n 6 -s 24 -r -c
ARG seed 24
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/z");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (z,1)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/z/t");

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (z,1)] [(.,1) (..,0) (t,2)] [] [] [] [] [] []

creat("/z/z");

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (z,1)] [(.,1) (..,0) (t,2) (z,3)] [] [] [] [] [] []

fd=open("/z/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:1] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (z,1)] [(.,1) (..,0) (t,2) (z,3)] [y] [] [] [] [] []


creat("/y");

inode bitmap  11111000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:1] [f a:-1 r:1] [] []
[]
data bitmap   11100000
data          [(.,0) (..,0) (z,1) (y,4)] [(.,1) (..,0) (t,2) (z,3)] [y] [] [] []
[] []
```

```
fd=open("/y", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11111000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:1] [f a:3 r:1] [] []
[]
data bitmap   11110000
data          [(.,0) (..,0) (z,1) (y,4)] [(.,1) (..,0) (t,2) (z,3)] [y] [v] [] []
[] []
```

分析上述情境，我们不难发现：分配算法会使用最近可分配的 inode 与数据块。

答案验证

seed 21

```
python2 vsfs.py -n 6 -s 21 -r -c
ARG seed 21
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/o");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (o,1)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/b");

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/o/q");

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3)] [] [] [] [] [] []

fd=open("/b", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:2 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11100000
```

```
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3)] [m] [] [] [] [] []


fd=open("/o/q", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:2 r:1] [f a:3 r:1] [] [] [] []
data bitmap   11110000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3)] [m] [j] [] [] [] []

creat("/o/j");

inode bitmap  11111000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:2 r:1] [f a:3 r:1] [f a:-1 r:1] [] []
[]
data bitmap   11110000
data          [(.,0) (..,0) (o,1) (b,2)] [(.,1) (..,0) (q,3) (j,4)] [m] [j] [] []
[] []
```

seed 22

```
python2 vsfs.py -n 6 -s 22 -r -c
ARG seed 22
ARG numInodes 8
ARG numData 8
ARG numRequests 6
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

creat("/z");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (z,1)] [] [] [] [] [] [] []

fd=open("/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (z,1)] [q] [] [] [] [] [] []

unlink("/z");

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []
```

```
creat("/y");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1)] [] [] [] [] [] [] []

link("/y", "/s");

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:2] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1) (s,1)] [] [] [] [] [] [] []

creat("/e");

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:-1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1) (s,1) (e,2)] [] [] [] [] [] [] []
```

经验证，答案正确。

Problem3

问题描述

现在将文件系统中的数据块数量减少到非常少（比如两个），并用 100 个左右的请求来运行模拟器。在这种高度约束的布局中，哪些类型的文件最终会出现在文件系统中？什么类型的操作会失败？

问题分析

首先弄清不同操作会使磁盘状态发生何种改变：

- mkdir() - 创建文件夹：修改inode位图，增加一个inode用来存放新目录元数据，向存放新目录的目录块中增加一个条目，修改data位图，增加一个数据块用于存放新目录的内容，更新相应inode中的引用计数
- creat() - 创建新的空文件：修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数
- open(), write(), close() - 将一个块添加到文件：修改data位图，增加一个数据块用于存放文件的新内容，修改inode中的数据块地址字段
- link() - 创建一个文件的硬链接：修改inode，增加其中的引用计数，在保存链接的目录块中增加一个条目
- unlink() - 删除一个硬链接 (如果 linkcnt==0,删除文件)：修改inode，减小其中的引用计数，在保存链接的目录块中删除一个条目，当引用计数减为0时，删除文件，释放inode、数据块，修改inode位图、data位图

不难发现mkdir()和open(), write(), close()需要数据块，而creat()、link()、unlink()则不需要。

问题解答

两个数据块太少了，很难看出问题，这里我们用三个数据块。

```
python2 vsfs.py -d 3 -c -n 100
ARG seed 0
ARG numInodes 8
ARG numData 3
ARG numRequests 100
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   100
data          [(.,0) (..,0)] [] []

mkdir("/g");

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   110
data          [(.,0) (..,0) (g,1)] [(.,1) (..,0)] []

creat("/q");

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   110
data          [(.,0) (..,0) (g,1) (q,2)] [(.,1) (..,0)] []

creat("/u");

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   110
data          [(.,0) (..,0) (g,1) (q,2) (u,3)] [(.,1) (..,0)] []

link("/u", "/x");

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:2] [] [] [] []
data bitmap   110
data          [(.,0) (..,0) (g,1) (q,2) (u,3) (x,3)] [(.,1) (..,0)] []

mkdir("/t");
File system out of data blocks; rerun with more via command-line flag?
```

PS：不知道是模拟器有点问题还是 *VSFS* 设计问题,最后一个数据块不能使用。

因为mkdir()和open(), write(), close()需要数据块，而creat()、link()、unlink()不需要数据块，所以mkdir()和open(), write(), close()操作会失败，creat()、link()、unlink()操作不会失败。

Problem4

问题描述

现在做同样的事情，但针对 inodes。只有非常少的 inode，什么类型的操作才能成功？哪些通常会失败？文件系统的最终状态可能是什么？

问题分析

首先弄清不同操作会使磁盘状态发生何种改变：

- mkdir() - 创建文件夹：修改inode位图，增加一个inode用来存放新目录元数据，向存放新目录的目录块中增加一个条目，修改data位图，增加一个数据块用于存放新目录的内容，更新相应inode中的引用计数
- creat() - 创建新的空文件：修改inode位图，增加一个inode用来存放新文件元数据，向存放新文件的目录块中增加一个条目，更新相应inode中的引用计数
- open(), write(), close() - 将一个块添加到文件：修改data位图，增加一个数据块用于存放文件的新内容，修改inode中的数据块地址字段
- link() - 创建一个文件的硬链接：修改inode，增加其中的引用计数，在保存链接的目录块中增加一个条目
- unlink() - 删除一个硬链接 (如果 linkcnt==0,删除文件)：修改inode，减小其中的引用计数，在保存链接的目录块中删除一个条目，当引用计数减为0时，删除文件，释放inode、数据块，修改inode位图、data位图

不难发现mkdir()和creat()需要inode，而open(), write(), close() 、link()、unlink()则不需要。

问题解答

```
python2 vsfs.py -i 3 -c -n 100
ARG seed 0
ARG numInodes 3
ARG numData 8
ARG numRequests 100
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  100
inodes        [d a:0 r:2] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/g");

inode bitmap  110
inodes        [d a:0 r:3] [d a:1 r:2] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1)] [(.,1) (..,0)] [] [] [] [] [] []

creat("/q");
File system out of inodes; rerun with more via command-line flag?
```

因为mkdir()和creat()需要inode，而open(), write(), close() 、link()、unlink()不需要inode，所以mkdir()和creat()操作会失败，open(), write(), close() 、link()、unlink()操作不会失败。本例中文件系统的最终状态是：

```
inode bitmap  110
inodes        [d a:0 r:3] [d a:1 r:2] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1)] [(.,1) (..,0)] [] [] [] [] [] []
```