File System Implementation

Annalise Tarhan

March 16, 2021

1 Run the simulator with some different random seeds and see if you can figure out which operations must have taken place between each state change.

Seed 17
mkdir(“/u”);
creat(“/a”);
unlink(“/a”);
mkdir(“/z”);
mkdir(“/s”);
creat(“/z/x”);
link(“/z/x”, “/u/b”);
unlink(“/u/b”);
fd=open(“/z/x”, O_WRONLY—O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
creat(“/u/b”);

Seed 18
mkdir(“/f”);
creat(“/s”);
mkdir(“/h);
fd=open(“/s”, O_WRONLY—O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
creat(“/f/o”);
creat(“/c”);
unlink(“/c”);
fd=open(“/f/o”, O_WRONLY—O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
unlink(“/s”); unlink(“/f/o”);

Seed 19
creat(“k”);
creat(“g”);
fd=open(“/k”, O_WRONLY—O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
link(“/k”, “b”);

```
link(“/k”, “t”);
unlink(“k”);
mkdir(“/r”);
mkdir(“/p”);
creat(“/r/d”);
link(“g”, “s”);
```

Seed 20

```
creat(“/x”);
fd=open(“/x”, O_WRONLY—O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
creat(“/k”);
creat(“/y”);
unlink(“/x”);
unlink(“/y”);
unlink(“/k”);
creat(“/p”);
fd=open(“/p”, O_WRONLY—O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
link(“/p”, “/s”);
```

2 Now do the same, using different random seeds, except run with the -r flag. What can you conclude about the inode and data-block allocation algorithms, in terms of which blocks they prefer to allocate?

The inode and data-block allocation algorithms choose the first available block.

Seed 21

```
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)][x][][][][][]

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)][x][y][][][][]

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)][x][y][][][][]

unlink(“b”);
inode bitmap 11011000
inodes [d a:0 r:3][d a:1 r:2][][f a:3 r:1][f a:-1 r:1][][][]
data bitmap 11010000
data [(.,0) (..,0) (o,1)][(.,1) (..,0) (q,3) (j,4)][][y][][][][]

fd=open(“/o/j”, O_WRONLY—O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
inode bitmap 11011000
inodes [d a:0 r:3][d a:1 r:2][][f a:3 r:1][f a:2 r:1][][][]
data bitmap 11110000
data [(.,0) (..,0) (o,1)][(.,1) (..,0) (q,3) (j,4)][z][y][][][][]

creat(“/o/x”);
inode bitmap 11111000
inodes [d a:0 r:3][d a:1 r:2][f a:-1 r:1][f a:3 r:1][f a:2 r:1][][][]
data bitmap 11110000
data [(.,0) (..,0) (o,1)][(.,1) (..,0) (q,3) (j,4) (x,2)][z][y][][][][]
```

```
mkdir("/o/t");
inode bitmap 11111100
inodes [d a:0 r:3][d a:1 r:3][f a:-1 r:1][f a:3 r:1][f a:2 r:1][d a:4 r:2][][]
data bitmap 11111000
data [(.,0) (..,0) (o,1)][(.,1) (..,0) (q,3) (j,4) (x,2) (t,5)][z][y][(.,5) (..,1)][][][]
```

3 Now reduce the number of data blocks in the file system to very low numbers and run the simulator for a hundred or so requests. What types of files end up in the file system in this highly-constrained layout? What types of operations fail?

The simulator aborts at the first failed operation, so the result is highly dependent on which operations are given first. The winners are the first few files and directories to be created, the losers are all the rest.

4 Now do the same, but with inodes. With very few inodes, what types of operations can succeed? Which will usually fail? What is the final state of the file system likely to be?

The result is the same. The first few files and directories are created and everything afterwards fails. If the simulation were to proceed, the operations that would succeed are those that created new files within existing directories, but efforts to write to them would fail.