Operationg Systems Midterm

Name – Surname : Ayşe Akışık - Betül Berna Soylu

Numbers : 171805007 – 171805019

Class : CSE306

Course : Operating Systems

Topic : Log – Structured File Systems

Date : 23.04.2021

QUESTIONS

Question-1) Run ./lfs.py -n 3, perhaps varying the seed (-s). Can you figure out which commands were run to generate the final file system contents? Can you tell which order those commands were issued?

In the "lfs.py -n 3" command, we say we will enter 3 parameters, but since we do not enter parameters, we cannot see which commands work.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -n 3

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

command?
command?
command?

FINAL file system contents:
[   0 ] ?    checkpoint: 14 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [ku3,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0
[   9 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- -- -- -- 8
[  10 ] ?    chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  11 ] ?    [.,0] [..,0] [ku3,1] [qg9,2] -- -- -- --
[  12 ] ?    type:dir size:1 refs:2 ptrs: 11 -- -- -- -- -- -- --
[  13 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  14 ] ?    chunk(imap): 12 9 13 -- -- -- -- -- -- -- -- -- -- -- -- --
```

Finally, can you determine the liveness of each block in the final file system state?

We can determine whether each block is alive or dead using the "-c" parameter.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -n 1 -o -c

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /ku3

FINAL file system contents:
[   0 ] live checkpoint: 7 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ]      [.,0] [..,0] -- -- -- -- -- --
[   2 ]      type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ]      chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] live [.,0] [..,0] [ku3,1] -- -- -- -- --
[   5 ] live type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] live type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] live chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

How much harder does the task become for you as you increase the number of commands issued (i.e., change -n 3 to -n 5)?

Block operations increase as we increase the number of parameters. Link file and "dir" file are created. Therefore, more work to be done. If the increment is too large, the code may not work.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -n 3 -n 5 -o

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /ku3
write file  /ku3 offset=7 size=4
create file /qg9
link file   /qg9 /is8
create dir  /cl6

FINAL file system contents:
[   0 ] ?    checkpoint: 23 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [ku3,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    x0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x0
[   9 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- -- -- -- 8
[  10 ] ?    chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- --
[  11 ] ?    [.,0] [..,0] [ku3,1] [qg9,2] -- -- -- --
[  12 ] ?    type:dir size:1 refs:2 ptrs: 11 -- -- -- -- -- -- --
[  13 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  14 ] ?    chunk(imap): 12 9 13 -- -- -- -- -- -- -- -- -- -- -- -- --
[  15 ] ?    [.,0] [..,0] [ku3,1] [qg9,2] [is8,2] -- -- --
[  16 ] ?    type:dir size:1 refs:2 ptrs: 15 -- -- -- -- -- -- --
[  17 ] ?    type:reg size:0 refs:2 ptrs: -- -- -- -- -- -- -- --
[  18 ] ?    chunk(imap): 16 9 17 -- -- -- -- -- -- -- -- -- -- -- -- --
[  19 ] ?    [.,0] [..,0] [ku3,1] [qg9,2] [is8,2] [cl6,3] -- --
[  20 ] ?    [.,3] [..,0] -- -- -- -- -- --
[  21 ] ?    type:dir size:1 refs:3 ptrs: 19 -- -- -- -- -- -- --
[  22 ] ?    type:dir size:1 refs:2 ptrs: 20 -- -- -- -- -- -- --
[  23 ] ?    chunk(imap): 21 9 17 22 -- -- -- -- -- -- -- -- -- -- -- --
```

-n 3 to -n 1000:

```
command?
command?
command?
Traceback (most recent call last):
  File "C:\Users\BS\Desktop\pythonProject\lfs.py", line 894, in <module>
    rc = L.file_link(command_and_args[1], command_and_args[2])
  File "C:\Users\BS\Desktop\pythonProject\lfs.py", line 528, in file_link
    dst_index_to_update, dst_parent_size, new_directory_block = self.__add_dir_entry(dst_parent_inode,
  File "C:\Users\BS\Desktop\pythonProject\lfs.py", line 408, in __add_dir_entry
    inode_index, dirblock_index = self.__find_matching_dir_slot('-', parent_inode)
  File "C:\Users\BS\Desktop\pythonProject\lfs.py", line 396, in __find_matching_dir_slot
    for inode_index in range(inode['size']):
TypeError: string indices must be integers
```

Question-2) If you find the above painful, you can help yourself a little bit by showing the set of updates caused by each specific command. To do so, run ./lfs.py -n 3 -i.

When it arrives at the checkpoint, the update process is repeated.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -n 3 -i

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

command?

[   0 ] ?    checkpoint: 7 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
...
[   4 ] ?    [.,0] [..,0] [ku3,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- --


command?

[   0 ] ?    checkpoint: 10 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
...
[   8 ] ?    z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0
[   9 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- -- -- -- 8
[  10 ] ?    chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- --


command?

[   0 ] ?    checkpoint: 14 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
...
[  11 ] ?    [.,0] [..,0] [ku3,1] [qg9,2] -- -- -- --
[  12 ] ?    type:dir size:1 refs:2 ptrs: 11 -- -- -- -- -- -- --
[  13 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  14 ] ?    chunk(imap): 12 9 13 -- -- -- -- -- -- -- -- -- -- -- -- --


FINAL file system contents:
[   0 ] ?    checkpoint: 14 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [ku3,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0
[   9 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- -- -- -- 8
[  10 ] ?    chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- --
[  11 ] ?    [.,0] [..,0] [ku3,1] [qg9,2] -- -- -- --
[  12 ] ?    type:dir size:1 refs:2 ptrs: 11 -- -- -- -- -- -- --
[  13 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  14 ] ?    chunk(imap): 12 9 13 -- -- -- -- -- -- -- -- -- -- -- -- --
```

Now see if it is easier to understand what each command must have been. Change the random seed to get different commands to interpret (e.g., -s 1, -s 2, -s 3, etc.).

As the number of seeds changes and increases, the files created and the procedures to do change. For example, the command "-s 2" changes the control point, the size of the file, how many times it is printed or when it is updated.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -s 2
INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
command?
command?
command?
FINAL file system contents:
[   0 ] ?    checkpoint: 17 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [vt6,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    y0y0y0y0y0y0y0y0y0y0y0y0y0y0y0y0
[   9 ] ?    o1o1o1o1o1o1o1o1o1o1o1o1o1o1o1o1
[  10 ] ?    l2l2l2l2l2l2l2l2l2l2l2l2l2l2l2l2
[  11 ] ?    g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3
[  12 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- 8 9 10 11
[  13 ] ?    chunk(imap): 5 12 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  14 ] ?    [.,0] [..,0] [vt6,1] [ks9,2] -- -- -- --
[  15 ] ?    type:dir size:1 refs:2 ptrs: 14 -- -- -- -- -- -- --
[  16 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  17 ] ?    chunk(imap): 15 12 16 -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -s 3

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

command?
command?
command?

FINAL file system contents:
[   0 ] ?    checkpoint: 15 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [jp6,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    [.,0] [..,0] [jp6,1] [vg2,2] -- -- -- --
[   9 ] ?    type:dir size:1 refs:2 ptrs: 8 -- -- -- -- -- -- --
[  10 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  11 ] ?    chunk(imap): 9 6 10 -- -- -- -- -- -- -- -- -- -- -- -- --
[  12 ] ?    [.,0] [..,0] [jp6,1] [vg2,2] [mq1,1] -- -- --
[  13 ] ?    type:dir size:1 refs:2 ptrs: 12 -- -- -- -- -- -- --
[  14 ] ?    type:reg size:0 refs:2 ptrs: -- -- -- -- -- -- -- --
[  15 ] ?    chunk(imap): 13 14 10 -- -- -- -- -- -- -- -- -- -- -- -- --
```

Question-3) To further test your ability to figure out what updates are made to disk by each command, run the following: ./lfs.py -o -F -s 100 (and perhaps a few other random seeds). This just shows a set of commands and does NOT show you the final state of the file system. Can you reason about what the final state of the file system must be?

- The size of the file is 7. This file has been written 4 times (offset).
- Therefore, the file starts and ends the process for the last time at the 12th checkpoint.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -o -F -s 100

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /us7
write file  /us7 offset=4 size=0
write file  /us7 offset=7 size=7
```

Question-4) Now see if you can determine which files and directories are live after a number of file and directory operations. Run tt ./lfs.py -n 20 -s 1 and then examine the final file system state.

We cannot determine if it is live with this command line.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -n 20 -s 1

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
command?
```

```
FINAL file system contents:
[   0 ] ?    checkpoint: 99 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [tg4,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    type:reg size:6 refs:1 ptrs: -- -- -- -- -- -- -- --
[   9 ] ?    chunk(imap): 5 8 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  10 ] ?    [.,0] [..,0] [tg4,1] [lt0,2] -- -- -- --
[  11 ] ?    type:dir size:1 refs:2 ptrs: 10 -- -- -- -- -- -- --
[  12 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  13 ] ?    chunk(imap): 11 8 12 -- -- -- -- -- -- -- -- -- -- -- -- --
[  14 ] ?    n0n0n0n0n0n0n0n0n0n0n0n0n0n0n0n0
[  15 ] ?    y1y1y1y1y1y1y1y1y1y1y1y1y1y1y1y1
[  16 ] ?    p2p2p2p2p2p2p2p2p2p2p2p2p2p2p2p2
[  17 ] ?    l3l3l3l3l3l3l3l3l3l3l3l3l3l3l3l3
[  18 ] ?    h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4
[  19 ] ?    o5o5o5o5o5o5o5o5o5o5o5o5o5o5o5o5
[  20 ] ?    y6y6y6y6y6y6y6y6y6y6y6y6y6y6y6y6
[  21 ] ?    type:reg size:8 refs:1 ptrs: -- 14 15 16 17 18 19 20
[  22 ] ?    chunk(imap): 11 8 21 -- -- -- -- -- -- -- -- -- -- -- -- --
[  23 ] ?    [.,0] [..,0] [tg4,1] [lt0,2] [oy3,1] -- -- --
[  24 ] ?    type:dir size:1 refs:2 ptrs: 23 -- -- -- -- -- -- --
[  25 ] ?    type:reg size:6 refs:2 ptrs: -- -- -- -- -- -- -- --
[  26 ] ?    chunk(imap): 24 25 21 -- -- -- -- -- -- -- -- -- -- -- -- --
[  27 ] ?    [.,0] [..,0] [tg4,1] [lt0,2] [oy3,1] [af4,3] -- --
[  28 ] ?    type:dir size:1 refs:2 ptrs: 27 -- -- -- -- -- -- --
[  29 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  30 ] ?    chunk(imap): 28 25 21 29 -- -- -- -- -- -- -- -- -- -- -- --
```

```
[  31 ] ?    a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0
[  32 ] ?    type:reg size:6 refs:2 ptrs: -- 31 -- -- -- -- -- --
[  33 ] ?    chunk(imap): 28 32 21 29 -- -- -- -- -- -- -- -- -- -- -- --
[  34 ] ?    u0u0u0u0u0u0u0u0u0u0u0u0u0u0u0u0
[  35 ] ?    v1v1v1v1v1v1v1v1v1v1v1v1v1v1v1v1
[  36 ] ?    x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2
[  37 ] ?    t3t3t3t3t3t3t3t3t3t3t3t3t3t3t3t3
[  38 ] ?    v4v4v4v4v4v4v4v4v4v4v4v4v4v4v4v4
[  39 ] ?    n5n5n5n5n5n5n5n5n5n5n5n5n5n5n5n5
[  40 ] ?    type:reg size:8 refs:1 ptrs: 34 35 36 37 38 39 19 20
[  41 ] ?    chunk(imap): 28 32 40 29 -- -- -- -- -- -- -- -- -- -- -- --
[  42 ] ?    o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0
[  43 ] ?    l1l1l1l1l1l1l1l1l1l1l1l1l1l1l1l1
[  44 ] ?    b2b2b2b2b2b2b2b2b2b2b2b2b2b2b2b2
[  45 ] ?    w3w3w3w3w3w3w3w3w3w3w3w3w3w3w3w3
[  46 ] ?    o4o4o4o4o4o4o4o4o4o4o4o4o4o4o4o4
[  47 ] ?    f5f5f5f5f5f5f5f5f5f5f5f5f5f5f5f5
[  48 ] ?    n6n6n6n6n6n6n6n6n6n6n6n6n6n6n6n6
[  49 ] ?    type:reg size:8 refs:2 ptrs: -- 42 43 44 45 46 47 48
[  50 ] ?    chunk(imap): 28 49 40 29 -- -- -- -- -- -- -- -- -- -- -- --
[  51 ] ?    [.,0] [..,0] -- [lt0,2] [oy3,1] [af4,3] -- --
[  52 ] ?    type:dir size:1 refs:2 ptrs: 51 -- -- -- -- -- -- --
[  53 ] ?    type:reg size:8 refs:1 ptrs: -- 42 43 44 45 46 47 48
[  54 ] ?    chunk(imap): 52 53 40 29 -- -- -- -- -- -- -- -- -- -- -- --
[  55 ] ?    m0m0m0m0m0m0m0m0m0m0m0m0m0m0m0m0
[  56 ] ?    j1j1j1j1j1j1j1j1j1j1j1j1j1j1j1j1
[  57 ] ?    i2i2i2i2i2i2i2i2i2i2i2i2i2i2i2i2
[  58 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- -- 55 56 57
[  59 ] ?    chunk(imap): 52 53 40 58 -- -- -- -- -- -- -- -- -- -- -- --
[  60 ] ?    a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0
[  61 ] ?    f1f1f1f1f1f1f1f1f1f1f1f1f1f1f1f1
[  62 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- -- 60 61 57
[  63 ] ?    chunk(imap): 52 53 40 62 -- -- -- -- -- -- -- -- -- -- -- --
```

```
[  67 ] ?    chunk(imap): 52 53 40 66 -- -- -- -- -- -- -- -- -- -- -- --
[  68 ] ?    u0u0u0u0u0u0u0u0u0u0u0u0u0u0u0u0
[  69 ] ?    v1v1v1v1v1v1v1v1v1v1v1v1v1v1v1v1
[  70 ] ?    g2g2g2g2g2g2g2g2g2g2g2g2g2g2g2g2
[  71 ] ?    v3v3v3v3v3v3v3v3v3v3v3v3v3v3v3v3
[  72 ] ?    r4r4r4r4r4r4r4r4r4r4r4r4r4r4r4r4
[  73 ] ?    c5c5c5c5c5c5c5c5c5c5c5c5c5c5c5c5
[  74 ] ?    type:reg size:8 refs:1 ptrs: 34 68 69 70 71 72 73 20
[  75 ] ?    chunk(imap): 52 53 74 66 -- -- -- -- -- -- -- -- -- -- -- --
[  76 ] ?    a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0
[  77 ] ?    a1a1a1a1a1a1a1a1a1a1a1a1a1a1a1a1
[  78 ] ?    t2t2t2t2t2t2t2t2t2t2t2t2t2t2t2t2
[  79 ] ?    g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3
[  80 ] ?    type:reg size:8 refs:1 ptrs: 34 68 69 70 76 77 78 79
[  81 ] ?    chunk(imap): 52 53 80 66 -- -- -- -- -- -- -- -- -- -- -- --
[  82 ] ?    [.,0] [..,0] [ln7,4] [lt0,2] [oy3,1] [af4,3] -- --
[  83 ] ?    [.,4] [..,0] -- -- -- -- -- --
[  84 ] ?    type:dir size:1 refs:3 ptrs: 82 -- -- -- -- -- -- --
[  85 ] ?    type:dir size:1 refs:2 ptrs: 83 -- -- -- -- -- -- --
[  86 ] ?    chunk(imap): 84 53 80 66 85 -- -- -- -- -- -- -- -- -- -- --
[  87 ] ?    type:reg size:8 refs:1 ptrs: -- 42 43 44 45 46 47 48
[  88 ] ?    chunk(imap): 84 87 80 66 85 -- -- -- -- -- -- -- -- -- -- --
[  89 ] ?    [.,4] [..,0] [zp3,5] -- -- -- -- --
[  90 ] ?    type:dir size:1 refs:2 ptrs: 89 -- -- -- -- -- -- --
[  91 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  92 ] ?    chunk(imap): 84 87 80 66 90 91 -- -- -- -- -- -- -- -- -- --
[  93 ] ?    [.,4] [..,0] [zp3,5] [zu5,6] -- -- -- --
[  94 ] ?    type:dir size:1 refs:2 ptrs: 93 -- -- -- -- -- -- --
[  95 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  96 ] ?    chunk(imap): 84 87 80 66 94 91 95 -- -- -- -- -- -- -- -- --
[  97 ] ?    [.,0] [..,0] [ln7,4] [lt0,2] -- [af4,3] -- --
[  98 ] ?    type:dir size:1 refs:3 ptrs: 97 -- -- -- -- -- -- --
[  99 ] ?    chunk(imap): 98 -- 80 66 94 91 95 -- -- -- -- -- -- -- -- --
```

Can you figure out which pathnames are valid? Run tt ./lfs.py -n 20 -s 1 -c -v to see the results. Run with -o to see if your answers match up given the series of random commands.

['/ln7'], ['/lt0', '/af4', '/ln7/zp3', 'ln7/zu5'] pathnames are valid.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -n 20 -s 1 -c -v -o

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /tg4
write file  /tg4 offset=6 size=0
create file /lt0
write file  /lt0 offset=1 size=7
link file   /tg4 /oy3
create file /af4
write file  /tg4 offset=1 size=1
write file  /lt0 offset=0 size=6
write file  /oy3 offset=1 size=7
delete file /tg4
write file  /af4 offset=5 size=7
write file  /af4 offset=5 size=2
write file  /af4 offset=6 size=4
write file  /lt0 offset=1 size=6
write file  /lt0 offset=4 size=5
create dir  /ln7
write file  /oy3 offset=3 size=0
create file /ln7/zp3
create file /ln7/zu5
delete file /oy3
```

```
FINAL file system contents:
[   0 ] live checkpoint: 99 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ]      [.,0] [..,0] -- -- -- -- -- --
[   2 ]      type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ]      chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ]      [.,0] [..,0] [tg4,1] -- -- -- -- --
[   5 ]      type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ]      type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ]      chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ]      type:reg size:6 refs:1 ptrs: -- -- -- -- -- -- -- --
[   9 ]      chunk(imap): 5 8 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  10 ]      [.,0] [..,0] [tg4,1] [lt0,2] -- -- -- --
[  11 ]      type:dir size:1 refs:2 ptrs: 10 -- -- -- -- -- -- --
[  12 ]      type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  13 ]      chunk(imap): 11 8 12 -- -- -- -- -- -- -- -- -- -- -- -- --
[  14 ]      n0n0n0n0n0n0n0n0n0n0n0n0n0n0n0n0
[  15 ]      y1y1y1y1y1y1y1y1y1y1y1y1y1y1y1y1
[  16 ]      p2p2p2p2p2p2p2p2p2p2p2p2p2p2p2p2
[  17 ]      l3l3l3l3l3l3l3l3l3l3l3l3l3l3l3l3
[  18 ]      h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4h4
[  19 ]      o5o5o5o5o5o5o5o5o5o5o5o5o5o5o5o5
[  20 ]      y6y6y6y6y6y6y6y6y6y6y6y6y6y6y6y6
[  21 ]      type:reg size:8 refs:1 ptrs: -- 14 15 16 17 18 19 20
[  22 ]      chunk(imap): 11 8 21 -- -- -- -- -- -- -- -- -- -- -- -- --
[  23 ]      [.,0] [..,0] [tg4,1] [lt0,2] [oy3,1] -- -- --
[  24 ]      type:dir size:1 refs:2 ptrs: 23 -- -- -- -- -- -- --
[  25 ]      type:reg size:6 refs:2 ptrs: -- -- -- -- -- -- -- --
[  26 ]      chunk(imap): 24 25 21 -- -- -- -- -- -- -- -- -- -- -- -- --
[  27 ]      [.,0] [..,0] [tg4,1] [lt0,2] [oy3,1] [af4,3] -- --
[  28 ]      type:dir size:1 refs:2 ptrs: 27 -- -- -- -- -- -- --
[  29 ]      type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  30 ]      chunk(imap): 28 25 21 29 -- -- -- -- -- -- -- -- -- -- -- --
```

```
[  31 ]      a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0
[  32 ]      type:reg size:6 refs:2 ptrs: -- 31 -- -- -- -- -- --
[  33 ]      chunk(imap): 28 32 21 29 -- -- -- -- -- -- -- -- -- -- -- --
[  34 ] live u0u0u0u0u0u0u0u0u0u0u0u0u0u0u0u0
[  35 ]      v1v1v1v1v1v1v1v1v1v1v1v1v1v1v1v1
[  36 ]      x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2
[  37 ]      t3t3t3t3t3t3t3t3t3t3t3t3t3t3t3t3
[  38 ]      v4v4v4v4v4v4v4v4v4v4v4v4v4v4v4v4
[  39 ]      n5n5n5n5n5n5n5n5n5n5n5n5n5n5n5n5
[  40 ]      type:reg size:8 refs:1 ptrs: 34 35 36 37 38 39 19 20
[  41 ]      chunk(imap): 28 32 40 29 -- -- -- -- -- -- -- -- -- -- -- --
[  42 ]      o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0o0
[  43 ]      l1l1l1l1l1l1l1l1l1l1l1l1l1l1l1l1
[  44 ]      b2b2b2b2b2b2b2b2b2b2b2b2b2b2b2b2
[  45 ]      w3w3w3w3w3w3w3w3w3w3w3w3w3w3w3w3
[  46 ]      o4o4o4o4o4o4o4o4o4o4o4o4o4o4o4o4
[  47 ]      f5f5f5f5f5f5f5f5f5f5f5f5f5f5f5f5
[  48 ]      n6n6n6n6n6n6n6n6n6n6n6n6n6n6n6n6
[  49 ]      type:reg size:8 refs:2 ptrs: -- 42 43 44 45 46 47 48
[  50 ]      chunk(imap): 28 49 40 29 -- -- -- -- -- -- -- -- -- -- -- --
[  51 ]      [.,0] [..,0] -- [lt0,2] [oy3,1] [af4,3] -- --
[  52 ]      type:dir size:1 refs:2 ptrs: 51 -- -- -- -- -- -- --
[  53 ]      type:reg size:8 refs:1 ptrs: -- 42 43 44 45 46 47 48
[  54 ]      chunk(imap): 52 53 40 29 -- -- -- -- -- -- -- -- -- -- -- --
[  55 ]      m0m0m0m0m0m0m0m0m0m0m0m0m0m0m0m0
[  56 ]      j1j1j1j1j1j1j1j1j1j1j1j1j1j1j1j1
[  57 ]      i2i2i2i2i2i2i2i2i2i2i2i2i2i2i2i2
[  58 ]      type:reg size:8 refs:1 ptrs: -- -- -- -- -- 55 56 57
[  59 ]      chunk(imap): 52 53 40 58 -- -- -- -- -- -- -- -- -- -- -- --
[  60 ] live a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0
[  61 ]      f1f1f1f1f1f1f1f1f1f1f1f1f1f1f1f1
[  62 ]      type:reg size:8 refs:1 ptrs: -- -- -- -- -- 60 61 57
[  63 ]      chunk(imap): 52 53 40 62 -- -- -- -- -- -- -- -- -- -- -- --
```

```
[  70 ] live g2g2g2g2g2g2g2g2g2g2g2g2g2g2g2g2
[  71 ]      v3v3v3v3v3v3v3v3v3v3v3v3v3v3v3v3
[  72 ]      r4r4r4r4r4r4r4r4r4r4r4r4r4r4r4r4
[  73 ]      c5c5c5c5c5c5c5c5c5c5c5c5c5c5c5c5
[  74 ]      type:reg size:8 refs:1 ptrs: 34 68 69 70 71 72 73 20
[  75 ]      chunk(imap): 52 53 74 66 -- -- -- -- -- -- -- -- -- -- -- --
[  76 ] live a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0
[  77 ] live a1a1a1a1a1a1a1a1a1a1a1a1a1a1a1a1
[  78 ] live t2t2t2t2t2t2t2t2t2t2t2t2t2t2t2t2
[  79 ] live g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3
[  80 ] live type:reg size:8 refs:1 ptrs: 34 68 69 70 76 77 78 79
[  81 ]      chunk(imap): 52 53 80 66 -- -- -- -- -- -- -- -- -- -- -- --
[  82 ]      [.,0] [..,0] [ln7,4] [lt0,2] [oy3,1] [af4,3] -- --
[  83 ]      [.,4] [..,0] -- -- -- -- -- --
[  84 ]      type:dir size:1 refs:3 ptrs: 82 -- -- -- -- -- -- --
[  85 ]      type:dir size:1 refs:2 ptrs: 83 -- -- -- -- -- -- --
[  86 ]      chunk(imap): 84 53 80 66 85 -- -- -- -- -- -- -- -- -- -- --
[  87 ]      type:reg size:8 refs:1 ptrs: -- 42 43 44 45 46 47 48
[  88 ]      chunk(imap): 84 87 80 66 85 -- -- -- -- -- -- -- -- -- -- --
[  89 ]      [.,4] [..,0] [zp3,5] -- -- -- -- --
[  90 ]      type:dir size:1 refs:2 ptrs: 89 -- -- -- -- -- -- --
[  91 ] live type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  92 ]      chunk(imap): 84 87 80 66 90 91 -- -- -- -- -- -- -- -- -- --
[  93 ] live [.,4] [..,0] [zp3,5] [zu5,6] -- -- -- --
[  94 ] live type:dir size:1 refs:2 ptrs: 93 -- -- -- -- -- -- --
[  95 ] live type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  96 ]      chunk(imap): 84 87 80 66 94 91 95 -- -- -- -- -- -- -- -- --
[  97 ] live [.,0] [..,0] [ln7,4] [lt0,2] -- [af4,3] -- --
[  98 ] live type:dir size:1 refs:3 ptrs: 97 -- -- -- -- -- -- --
[  99 ] live chunk(imap): 98 -- 80 66 94 91 95 -- -- -- -- -- -- -- -- --

Live directories:  ['/ln7']
Live files:  ['/lt0', '/af4', '/ln7/zp3', '/ln7/zu5']
```

Use different random seeds to get more problems.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -n 20 -s 2 -c -v -o

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /vt6
write file  /vt6 offset=4 size=4
create file /ks9
link file   /ks9 /ga0
write file  /vt6 offset=3 size=7
write file  /ks9 offset=1 size=0
write file  /vt6 offset=4 size=7
create dir  /xu7
link file   /ga0 /xu7/jz9
create file /xu7/sl5
write file  /xu7/sl5 offset=4 size=6
write file  /xu7/sl5 offset=7 size=3
write file  /xu7/sl5 offset=5 size=3
create file /se9
create file /xu7/iy7
write file  /xu7/jz9 offset=5 size=4
write file  /ks9 offset=4 size=7
create dir  /vs9
create file /vs9/bq2
create file /vs9/cn8
```

```
Live directories:  ['/xu7', '/vs9']
Live files:  ['/vt6', '/ks9', '/ga0', '/xu7/jz9', '/xu7/sl5', '/se9', '/xu7/iy7', '/vs9/bq2', '/vs9/cn8']
```

Question-5) Now let's issue some specific commands. First, let's create a file and write to it repeatedly. To do so, use the -L flag, which lets you specify specific commands to execute. Let's create the file "/foo" and write to it four times: -L c,/foo:w,/foo,0,1:w,/foo,1,1:w,/foo,2,1:w,/foo,3,1 -o.

We did write 4 times without changing the control point and size of the file.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -L c,/foo:w,/foo,0,1:w,/foo,1,1:w,/foo,2,1:w,/foo,3,1 -o

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /foo
write file  /foo offset=0 size=1
write file  /foo offset=1 size=1
write file  /foo offset=2 size=1
write file  /foo offset=3 size=1

FINAL file system contents:
[   0 ] ?    checkpoint: 19 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0
[   9 ] ?    type:reg size:1 refs:1 ptrs: 8 -- -- -- -- -- -- --
[  10 ] ?    chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  11 ] ?    t0t0t0t0t0t0t0t0t0t0t0t0t0t0t0t0
[  12 ] ?    type:reg size:2 refs:1 ptrs: 8 11 -- -- -- -- -- --
[  13 ] ?    chunk(imap): 5 12 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  14 ] ?    k0k0k0k0k0k0k0k0k0k0k0k0k0k0k0k0
[  15 ] ?    type:reg size:3 refs:1 ptrs: 8 11 14 -- -- -- -- --
[  16 ] ?    chunk(imap): 5 15 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  17 ] ?    g0g0g0g0g0g0g0g0g0g0g0g0g0g0g0g0
[  18 ] ?    type:reg size:4 refs:1 ptrs: 8 11 14 17 -- -- -- --
[  19 ] ?    chunk(imap): 5 18 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

See if you can determine the liveness of the final file system state; use -c to check your answers.

We can see that the created files are live.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -L c,/foo:w,/foo,0,1:w,/foo,1,1:w,/foo,2,1:w,/foo,3,1 -o -c

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /foo
write file  /foo offset=0 size=1
write file  /foo offset=1 size=1
write file  /foo offset=2 size=1
write file  /foo offset=3 size=1

FINAL file system contents:
[   0 ] live checkpoint: 19 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ]      [.,0] [..,0] -- -- -- -- -- --
[   2 ]      type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ]      chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] live [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] live type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ]      type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ]      chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] live v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0
[   9 ]      type:reg size:1 refs:1 ptrs: 8 -- -- -- -- -- -- --
[  10 ]      chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- --
[  11 ] live t0t0t0t0t0t0t0t0t0t0t0t0t0t0t0t0
[  12 ]      type:reg size:2 refs:1 ptrs: 8 11 -- -- -- -- -- --
[  13 ]      chunk(imap): 5 12 -- -- -- -- -- -- -- -- -- -- -- -- --
[  14 ] live k0k0k0k0k0k0k0k0k0k0k0k0k0k0k0k0
[  15 ]      type:reg size:3 refs:1 ptrs: 8 11 14 -- -- -- -- --
[  16 ]      chunk(imap): 5 15 -- -- -- -- -- -- -- -- -- -- -- -- --
[  17 ] live g0g0g0g0g0g0g0g0g0g0g0g0g0g0g0g0
[  18 ] live type:reg size:4 refs:1 ptrs: 8 11 14 17 -- -- -- --
[  19 ] live chunk(imap): 5 18 -- -- -- -- -- -- -- -- -- -- -- -- --
```

Question-6) Now, let's do the same thing, but with a single write operation instead of four. Run ./lfs.py -o -L c,/foo:w,/foo,0,4 to create file "/foo" and write 4 blocks with a single write operation.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -o -L c,/foo:w,/foo,0,4

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /foo
write file  /foo offset=0 size=4

FINAL file system contents:
[   0 ] ?    checkpoint: 13 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0
[   9 ] ?    t1t1t1t1t1t1t1t1t1t1t1t1t1t1t1t1
[  10 ] ?    k2k2k2k2k2k2k2k2k2k2k2k2k2k2k2k2
[  11 ] ?    g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3g3
[  12 ] ?    type:reg size:4 refs:1 ptrs: 8 9 10 11 -- -- -- --
[  13 ] ?    chunk(imap): 5 12 -- -- -- -- -- -- -- -- -- -- -- -- --
```

Compute the liveness again, and check if you are right with -c.

We can reach group of blocks that are live. All the rest are dead and could be used again.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -c

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

command?
command?
command?

FINAL file system contents:
[   0 ] live checkpoint: 14 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ]      [.,0] [..,0] -- -- -- -- -- --
[   2 ]      type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ]      chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ]      [.,0] [..,0] [ku3,1] -- -- -- -- --
[   5 ]      type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ]      type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ]      chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] live z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0
[   9 ] live type:reg size:8 refs:1 ptrs: -- -- -- -- -- -- -- 8
[  10 ]      chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  11 ] live [.,0] [..,0] [ku3,1] [qg9,2] -- -- -- --
[  12 ] live type:dir size:1 refs:2 ptrs: 11 -- -- -- -- -- -- --
[  13 ] live type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  14 ] live chunk(imap): 12 9 13 -- -- -- -- -- -- -- -- -- -- -- -- --
```

What is the main difference between writing a file all at once (as we do here) versus doing it one block at a time (as above)? What does this tell you about the importance of buffering updates inmain memory as the real LFS does?

- When write a file all at once, the size will be 4.
- When write 4 files in a single block, the size will be 1.

This means that keeping 4 writes in one block takes up less space on the disk.

Question-7) Let's do another specific example. First, run the following: ./lfs.py -L c,/foo:w,/foo,0,1. What does this set of commands do?

One size write operation has occurred on the 1st inode. When I ran this command the initial size was 1. After writing, became offset = 0, size = 1.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -L c,/foo:w,/foo,0,1 -o

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /foo
write file  /foo offset=0 size=1

FINAL file system contents:
[   0 ] ?    checkpoint: 10 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0
[   9 ] ?    type:reg size:1 refs:1 ptrs: 8 -- -- -- -- -- -- --
[  10 ] ?    chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Now, run ./lfs.py -L c,/foo:w,/foo,7,1. What does this set of commands do?

Eight size write operation has occurred on the eighth inode. When I ran this command the initial size was 1. After writing, became offset = 7, size = 8.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -L c,/foo:w,/foo,7,1 -o

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /foo
write file  /foo offset=7 size=1

FINAL file system contents:
[   0 ] ?    checkpoint: 10 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0v0
[   9 ] ?    type:reg size:8 refs:1 ptrs: -- -- -- -- -- -- -- 8
[  10 ] ?    chunk(imap): 5 9 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

How are the two different? What can you tell about the size field in the inode from these two sets of commands?

In the first command set it is size 1, in the second command set it is size 8.

Question-8) Now let's look explicitly at file creation versus directory creation. Run simulations ./lfs.py -L c,/foo and ./lfs.py -L d,/foo to create a file and then a directory. What is similar about these runs, and what is different?

Similarities: They are on the same inode. ([foo, 1])

The size of the file and directory is the same.

Differences: Create types are different. (file – directory)

The directory was added above the created file.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -L c,/foo

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

command?

FINAL file system contents:
[   0 ] ?    checkpoint: 7 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -L d,/foo

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

command?

FINAL file system contents:
[   0 ] ?    checkpoint: 8 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    [.,1] [..,0] -- -- -- -- -- --
[   6 ] ?    type:dir size:1 refs:3 ptrs: 4 -- -- -- -- -- -- --
[   7 ] ?    type:dir size:1 refs:2 ptrs: 5 -- -- -- -- -- -- --
[   8 ] ?    chunk(imap): 6 7 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Question-9) The LFS simulator supports hard links as well. Run the following to study how they work: ./lfs.py -L c,/foo:l,/foo,/bar:l,/foo,/goo -o -i.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -L c,/foo:l,/foo,/bar:l,/foo,/goo -o -i

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /foo

[   0 ] ?    checkpoint: 7 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
...
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --


link file   /foo /bar

[   0 ] ?    checkpoint: 11 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
...
[   8 ] ?    [.,0] [..,0] [foo,1] [bar,1] -- -- -- --
[   9 ] ?    type:dir size:1 refs:2 ptrs: 8 -- -- -- -- -- -- --
[  10 ] ?    type:reg size:0 refs:2 ptrs: -- -- -- -- -- -- -- --
[  11 ] ?    chunk(imap): 9 10 -- -- -- -- -- -- -- -- -- -- -- -- -- --


link file   /foo /goo

[   0 ] ?    checkpoint: 15 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
...
[  12 ] ?    [.,0] [..,0] [foo,1] [bar,1] [goo,1] -- -- --
[  13 ] ?    type:dir size:1 refs:2 ptrs: 12 -- -- -- -- -- -- --
[  14 ] ?    type:reg size:0 refs:3 ptrs: -- -- -- -- -- -- -- --
[  15 ] ?    chunk(imap): 13 14 -- -- -- -- -- -- -- -- -- -- -- -- -- --



FINAL file system contents:
[   0 ] ?    checkpoint: 15 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [foo,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    [.,0] [..,0] [foo,1] [bar,1] -- -- -- --
[   9 ] ?    type:dir size:1 refs:2 ptrs: 8 -- -- -- -- -- -- --
[  10 ] ?    type:reg size:0 refs:2 ptrs: -- -- -- -- -- -- -- --
[  11 ] ?    chunk(imap): 9 10 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  12 ] ?    [.,0] [..,0] [foo,1] [bar,1] [goo,1] -- -- --
[  13 ] ?    type:dir size:1 refs:2 ptrs: 12 -- -- -- -- -- -- --
[  14 ] ?    type:reg size:0 refs:3 ptrs: -- -- -- -- -- -- -- --
[  15 ] ?    chunk(imap): 13 14 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

What blocks are written out when a hard link is created?

Checkpoint block

Block containing [name, inode number]

Block containing type, size and reference of link

Block specifying its current location on disk for each inode number (chunk(imap))

How is this similar to just creating a new file, and how is it different? How does the reference count field change as links are created?

The links have been created inside the file. Reference numbers increased one at a time.

Question-10) LFS makes many different policy decisions. We do not explore many of themhere – perhaps something left for the future – but here is a simple onewe do explore: the choice of inode number. First, run ./lfs.py -p c100 -n 10 -o -a s to show the usual behavior with the ”sequential” allocation policy, which tries to use free inode numbers nearest to zero.

Here the files were added sequentially and regularly. It take up less space on the disk as it was added sequentially.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -p c100 -n 10 -o -a s

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /kg5
create file /hm5
create file /ht6
create file /zv9
create file /xr4
create file /px9
create file /gu5
create file /kv6
create file /wg3
create file /og9

FINAL file system contents:
[   0 ] ?    checkpoint: 43 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [kg5,1] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    [.,0] [..,0] [kg5,1] [hm5,2] -- -- -- --
[   9 ] ?    type:dir size:1 refs:2 ptrs: 8 -- -- -- -- -- -- --
[  10 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  11 ] ?    chunk(imap): 9 6 10 -- -- -- -- -- -- -- -- -- -- -- --
[  12 ] ?    [.,0] [..,0] [kg5,1] [hm5,2] [ht6,3] -- -- --
[  13 ] ?    type:dir size:1 refs:2 ptrs: 12 -- -- -- -- -- -- --
[  14 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  15 ] ?    chunk(imap): 13 6 10 14 -- -- -- -- -- -- -- -- -- -- -- --
[  16 ] ?    [.,0] [..,0] [kg5,1] [hm5,2] [ht6,3] [zv9,4] -- --
[  17 ] ?    type:dir size:1 refs:2 ptrs: 16 -- -- -- -- -- -- --
[  18 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  19 ] ?    chunk(imap): 17 6 10 14 18 -- -- -- -- -- -- -- -- -- -- --
[  20 ] ?    [.,0] [..,0] [kg5,1] [hm5,2] [ht6,3] [zv9,4] [xr4,5] --
[  21 ] ?    type:dir size:1 refs:2 ptrs: 20 -- -- -- -- -- -- --
[  22 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  23 ] ?    chunk(imap): 21 6 10 14 18 22 -- -- -- -- -- -- -- -- -- --
[  24 ] ?    [.,0] [..,0] [kg5,1] [hm5,2] [ht6,3] [zv9,4] [xr4,5] [px9,6]
```

```
[  24 ] ?    [.,0] [..,0] [kg5,1] [hm5,2] [ht6,3] [zv9,4] [xr4,5] [px9,6]
[  25 ] ?    type:dir size:1 refs:2 ptrs: 24 -- -- -- -- -- -- --
[  26 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  27 ] ?    chunk(imap): 25 6 10 14 18 22 26 -- -- -- -- -- -- -- -- --
[  28 ] ?    [gu5,7] -- -- -- -- -- -- --
[  29 ] ?    type:dir size:2 refs:2 ptrs: 24 28 -- -- -- -- -- --
[  30 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  31 ] ?    chunk(imap): 29 6 10 14 18 22 26 30 -- -- -- -- -- -- -- --
[  32 ] ?    [gu5,7] [kv6,8] -- -- -- -- -- --
[  33 ] ?    type:dir size:2 refs:2 ptrs: 24 32 -- -- -- -- -- --
[  34 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  35 ] ?    chunk(imap): 33 6 10 14 18 22 26 30 34 -- -- -- -- -- -- --
[  36 ] ?    [gu5,7] [kv6,8] [wg3,9] -- -- -- -- --
[  37 ] ?    type:dir size:2 refs:2 ptrs: 24 36 -- -- -- -- -- --
[  38 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  39 ] ?    chunk(imap): 37 6 10 14 18 22 26 30 34 38 -- -- -- -- -- --
[  40 ] ?    [gu5,7] [kv6,8] [wg3,9] [og9,10] -- -- -- --
[  41 ] ?    type:dir size:2 refs:2 ptrs: 24 40 -- -- -- -- -- --
[  42 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  43 ] ?    chunk(imap): 41 6 10 14 18 22 26 30 34 38 42 -- -- -- -- --
```

Then, change to a ”random” policy by running ./lfs.py -p c100 -n 10 -o -a r (the -p c100 flag ensures 100 percent of the random operations are file creations).

Files take up more disk space because they are added scattered.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -p c100 -n 10 -o -a r

INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

create file /kg5
create file /hm5
create file /ht6
create file /zv9
create file /xr4
create file /px9
create file /gu5
create file /kv6
create file /wg3
create file /og9

FINAL file system contents:
[   0 ] ?    checkpoint: 52 38 -- -- -- 23 -- 13 53 -- -- 48 8 -- 33 --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [kg5,205] -- -- -- -- --
[   5 ] ?    type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ] ?    chunk(imap): 5 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   8 ] ?    chunk(imap): -- -- -- -- -- -- -- -- -- -- -- -- -- 6 -- --
[   9 ] ?    [.,0] [..,0] [kg5,205] [hm5,114] -- -- -- --
[  10 ] ?    type:dir size:1 refs:2 ptrs: 9 -- -- -- -- -- -- --
[  11 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  12 ] ?    chunk(imap): 10 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  13 ] ?    chunk(imap): -- -- 11 -- -- -- -- -- -- -- -- -- -- -- -- --
[  14 ] ?    [.,0] [..,0] [kg5,205] [hm5,114] [ht6,20] -- -- --
[  15 ] ?    type:dir size:1 refs:2 ptrs: 14 -- -- -- -- -- -- --
[  16 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  17 ] ?    chunk(imap): 15 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  18 ] ?    chunk(imap): -- -- -- -- 16 -- -- -- -- -- -- -- -- -- -- --
[  19 ] ?    [.,0] [..,0] [kg5,205] [hm5,114] [ht6,20] [zv9,81] -- --
[  20 ] ?    type:dir size:1 refs:2 ptrs: 19 -- -- -- -- -- -- --
[  21 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  22 ] ?    chunk(imap): 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  23 ] ?    chunk(imap): -- 21 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  24 ] ?    [.,0] [..,0] [kg5,205] [hm5,114] [ht6,20] [zv9,81] [xr4,130] --
[  25 ] ?    type:dir size:1 refs:2 ptrs: 24 -- -- -- -- -- -- --
```

```
[  25 ] ?    type:dir size:1 refs:2 ptrs: 24 -- -- -- -- -- -- --
[  26 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  27 ] ?    chunk(imap): 25 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  28 ] ?    chunk(imap): -- -- 26 -- -- -- -- -- -- -- -- -- -- -- -- --
[  29 ] ?    [.,0] [..,0] [kg5,205] [hm5,114] [ht6,20] [zv9,81] [xr4,130] [px9,238]
[  30 ] ?    type:dir size:1 refs:2 ptrs: 29 -- -- -- -- -- -- --
[  31 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  32 ] ?    chunk(imap): 30 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  33 ] ?    chunk(imap): -- -- -- -- -- -- -- -- -- -- -- -- -- -- 31 --
[  34 ] ?    [gu5,27] -- -- -- -- -- -- --
[  35 ] ?    type:dir size:2 refs:2 ptrs: 29 34 -- -- -- -- -- --
[  36 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  37 ] ?    chunk(imap): 35 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  38 ] ?    chunk(imap): -- -- -- -- 16 -- -- -- -- -- -- 36 -- -- -- --
[  39 ] ?    [gu5,27] [kv6,141] -- -- -- -- -- --
[  40 ] ?    type:dir size:2 refs:2 ptrs: 29 39 -- -- -- -- -- --
[  41 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  42 ] ?    chunk(imap): 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  43 ] ?    chunk(imap): -- -- 26 -- -- -- -- -- -- -- -- -- -- 41 -- --
[  44 ] ?    [gu5,27] [kv6,141] [wg3,180] -- -- -- -- --
[  45 ] ?    type:dir size:2 refs:2 ptrs: 29 44 -- -- -- -- -- --
[  46 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  47 ] ?    chunk(imap): 45 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  48 ] ?    chunk(imap): -- -- -- -- 46 -- -- -- -- -- -- -- -- -- -- --
[  49 ] ?    [gu5,27] [kv6,141] [wg3,180] [og9,140] -- -- -- --
[  50 ] ?    type:dir size:2 refs:2 ptrs: 29 49 -- -- -- -- -- --
[  51 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  52 ] ?    chunk(imap): 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[  53 ] ?    chunk(imap): -- -- 26 -- -- -- -- -- -- -- -- -- 51 41 -- --
```

What on-disk differences does a random policy versus a sequential policy result in? What does this say about the importance of choosing inode numbers in a real LFS?

"Sequential" allocation policy is less take up on the disk because files are placed in inodes sequentially. So, the random policy takes up more space.

Question-11) One last thing we've been assuming is that the LFS simulator always updates the checkpoint region after each update. In the real LFS, that isn't the case: it is updated periodically to avoid long seeks. Run ./lfs.py -N -i -o -s 1000 to see some operations and the intermediate and final states of the file system when the checkpoint region isn't forced to disk.

```
C:\Users\BS\Desktop\pythonProject>python lfs.py -N -i -o -s 1000
INITIAL file system contents:
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
create dir  /jm5
[   4 ] ?    [.,0] [..,0] [jm5,1] -- -- -- -- --
[   5 ] ?    [.,1] [..,0] -- -- -- -- -- --
[   6 ] ?    type:dir size:1 refs:3 ptrs: 4 -- -- -- -- -- -- --
[   7 ] ?    type:dir size:1 refs:2 ptrs: 5 -- -- -- -- -- -- --
[   8 ] ?    chunk(imap): 6 7 -- -- -- -- -- -- -- -- -- -- -- -- -- --
create file /jm5/jm2
[   9 ] ?    [.,1] [..,0] [jm2,2] -- -- -- -- --
[  10 ] ?    type:dir size:1 refs:2 ptrs: 9 -- -- -- -- -- -- --
[  11 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  12 ] ?    chunk(imap): 6 10 11 -- -- -- -- -- -- -- -- -- -- -- -- --
create dir  /lb9
[  13 ] ?    [.,0] [..,0] [jm5,1] [lb9,3] -- -- -- --
[  14 ] ?    [.,3] [..,0] -- -- -- -- -- --
[  15 ] ?    type:dir size:1 refs:4 ptrs: 13 -- -- -- -- -- -- --
[  16 ] ?    type:dir size:1 refs:2 ptrs: 14 -- -- -- -- -- -- --
[  17 ] ?    chunk(imap): 15 10 11 16 -- -- -- -- -- -- -- -- -- -- -- --
FINAL file system contents:
[   0 ] ?    checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] ?    [.,0] [..,0] -- -- -- -- -- --
[   2 ] ?    type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] ?    chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ] ?    [.,0] [..,0] [jm5,1] -- -- -- -- --
[   5 ] ?    [.,1] [..,0] -- -- -- -- -- --
[   6 ] ?    type:dir size:1 refs:3 ptrs: 4 -- -- -- -- -- -- --
[   7 ] ?    type:dir size:1 refs:2 ptrs: 5 -- -- -- -- -- -- --
[   8 ] ?    chunk(imap): 6 7 -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   9 ] ?    [.,1] [..,0] [jm2,2] -- -- -- -- --
[  10 ] ?    type:dir size:1 refs:2 ptrs: 9 -- -- -- -- -- -- --
[  11 ] ?    type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[  12 ] ?    chunk(imap): 6 10 11 -- -- -- -- -- -- -- -- -- -- -- -- --
[  13 ] ?    [.,0] [..,0] [jm5,1] [lb9,3] -- -- -- --
[  14 ] ?    [.,3] [..,0] -- -- -- -- -- --
[  15 ] ?    type:dir size:1 refs:4 ptrs: 13 -- -- -- -- -- -- --
[  16 ] ?    type:dir size:1 refs:2 ptrs: 14 -- -- -- -- -- -- --
[  17 ] ?    chunk(imap): 15 10 11 16 -- -- -- -- -- -- -- -- -- -- -- --
```

What would happen if the checkpoint region is never updated? What if it is updated periodically? Could you figure out how to recover the file system to the latest state by rolling forward in the log?

The checkpoint is not updated at all, which means that there are hardly any updates in the file.

This means that it is not possible to determine whether the file is dead or alive.

When updated periodically, we can see that the blocks are dead or alive, although there is no activity in the file.

Computer Specifications (We got the same results on both computers)

- 8gb RAM, i3 processor, 64 bit operating system, Windows10
- 4gb RAM, i5 processor, 64 bit operating system, Windows10