File system implementation: access

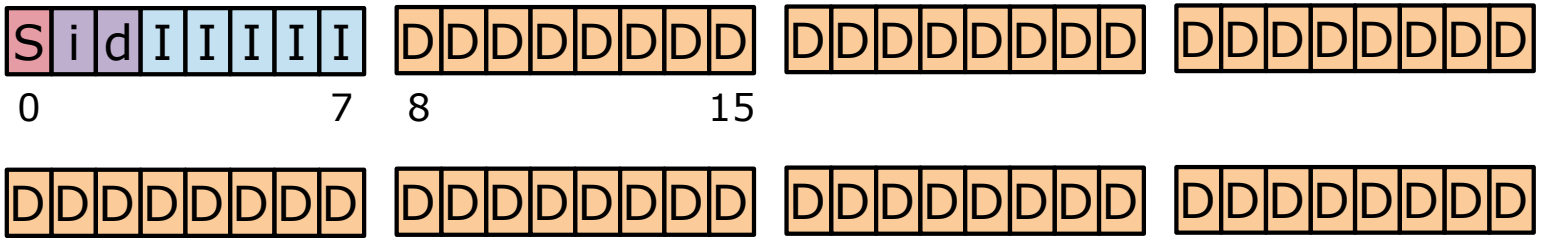
Glenn Bruns
CSUMB

Lecture Objectives

After this lecture, you should be able to:

- explain how to perform access operations on a simple file system
- hand-simulate the operations

Question: how to read file /foo/bar?



Hints:

- inode number of root directory is 0 (in most Unix file systems it's 2)

How to read /foo/bar

1. find inode for root directory (root inode # is 0)
2. use first direct pointer to get first data block
3. search for entry foo
4. find inode for foo and get first data block
5. search for entry bar
6. find bar’s inode, check permissions, get first direct pointer
7. read first block
8. update ‘last accessed time’ in inode for bar
9. read inode for bar and get second direct pointer
10. read second block
11. update last accessed time in inode for bar
12. etc.

VSFS simulation

We'll get a better feeling for this using the OSTEP simulator.

The VSFS file system is shown like this:

```
inode bitmap  11100000
inodes        [d a:0 r:4] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1) (z,2)] [u] [] [] [] [] [] []
```

There are 8 inodes, and 8 data blocks.

The unused inodes and data blocks are written as []

inodes

```
inode bitmap  11100000
inodes        [d a:0 r:4] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1) (z,2)] [u] [] [] [] [] [] []
```

An inode has the form

[type a:block-address r:ref-count]

type: “f” (regular file) or “d” (directory)

block-address: an index into data, or -1 if empty

ref-count: number of hard links to this (if regular file), or number of entries (if directory)

data blocks

```
inode bitmap   11100000
inodes         [d a:0 r:4] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap    11000000
data           [(.,0) (..,0) (y,1) (z,2)] [u] [] [] [] [] [] []
```

A data block has one of two forms

[a]

[(name,inode number),…]

The first form is the data for a regular file. (The letter is not meaningful.)

The second form is the data for a directory.

Interpreting the data

```
inode bitmap   11000000
inodes         [d a:0 r:4] [f a:1 r:2] [] [] [] [] [] []
data bitmap    11000000
data           [(.,0) (..,0) (y,1) (m,1)] [u] [] [] [] [] []
```

This file system contains:

- root directory, with files ., .., y, m
- file /y has contents ‘u’
- file /m is a hard link to /y

Exercise

```
inode bitmap   11100000
inodes         [d a:0 r:6] [f a:1 r:3] [f a:-1 r:1] [] [] [] [] []
data bitmap    11000000
data           [(.,0) (..,0) (p,1) (s,1) (a,2) (d,1)] [e] [] [] [] [] [] []
```

What's in this file system?

Which operation took place?

In default mode, the simulator asks you to identify the operation that must have occurred

```
Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:3] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (x,1)] [] [] [] [] [] [] []
```

The possible operations are: (quoted from the documentation)

- mkdir() creates a new directory
- creat() creates a new (empty) file
- open(), write(), close() appends a block to a file
- link() creates a hard link to a file
- unlink() unlinks a file (removing it if linkcnt==0)

Exercise

```
$ ./vsfs.py -s 23
…
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) (c,1)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

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

Showing correct output with -c

```
$ ./vsfs.py -s 23 -c | m
…
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:3] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (c,1)] [(.,1) (..,0) (t,2)] [] [] [] [] [] []
```

What is the resulting state?

In “reverse” mode (-r), the simulator gives you an operator and asks you to write the state

```
Initial state

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

creat("/x");

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

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

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

Exercise

```
Initial state

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

creat("/x");

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

Summary

We learned to perform file operations on the very simple file system (VSFS):

- create a directory
- create a file
- append to a file
- create a hard link to a file
- unlink a file