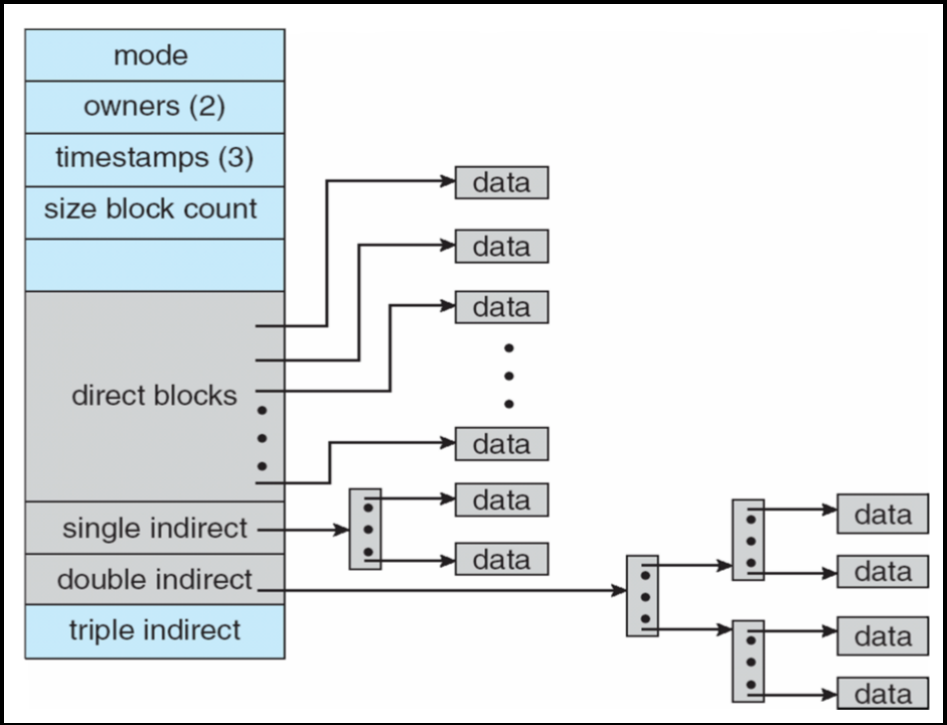
inodes

2024 Fall ECE 344: Operating Systems
Jon Eyolfson

Lecture 28
2.0.0

An inode Describes a File System Object (Files and Directories)



inodes Aim to Be Efficient for Small Files, and Support Large Ones

UNIX inodes hold metadata and pointers to blocks

Smaller files only use direct pointers

Larger files have additional index nodes with pointers to additional blocks

Very small files can have its contents directly inside the inode

inode Problem

Assume this scenario:

- An index block stores 12 direct pointers, 1 single, double and triple indirect pointer each
- A disk block is 8 KiB in size
- A pointer to a block is 4 Bytes
- Indirect blocks consist of direct pointers only

What is the maximum size of a file managed by this index block?

inode Solution

Assume this scenario:

- An index block stores 12 direct pointers, 1 single, double and triple indirect pointer each
- A disk block is 8 KiB in size
- A pointer to a block is 4 Bytes
- Indirect blocks consist of direct pointers only

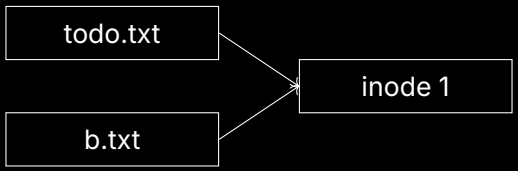
pointers per indirect table = $\frac{2^{13}}{2^2} = 2^{11}$
addressable blocks = $12 + 2^{11} + (2^{11})^2 + (2^{11})^3 \approx (2^{11})^3 = 2^{33}$
Total of bytes = $2^{33} * 2^{13} = 2^{46} = 64\text{TiB}$

Hard Links Are Pointers to inodes

A directory entry (aka filename) is called a *hard link*

A hard link points to one inode

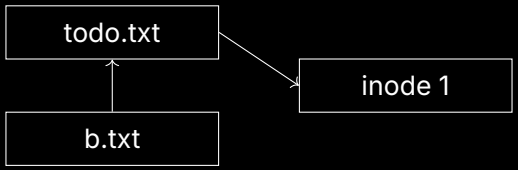
Multiple Hard Links Can Point to the Same inode



Deleting a file only removes a hard link
(the file can be hard liked somewhere else)

POSIX has the `unlink` call rather than a delete call

Soft Links Are Paths to Another File



When resolving the file, the file system is redirected somewhere else, so:

- Soft link targets do not need to exist
- Soft link targets can be deleted without notice of the soft link
- Unresolvable soft links lead to an exception

Filesystem Example Problem

```
touch todo.txt
ln todo.txt b.txt
ln -s todo.txt c.txt
mv todo.txt d.txt
rm b.txt
```

How does the FS look like before and after the mv and rm commands?

Filesystem Example Solution (1)

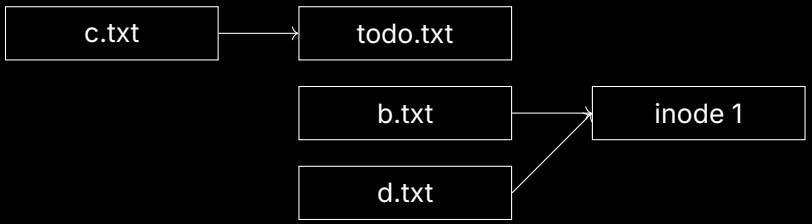
```
touch todo.txt
ln todo.txt b.txt
ln -s todo.txt c.txt
mv todo.txt d.txt
rm b.txt
```

Before mv:

Filesystem Example Solution (2)

```
touch todo.txt
ln todo.txt b.txt
ln -s todo.txt c.txt
mv todo.txt d.txt
rm b.txt
```

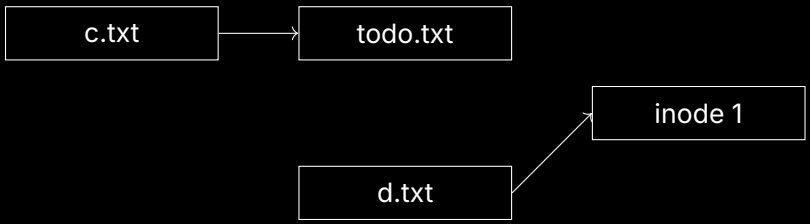
After mv:



Filesystem Example Solution (3)

```
touch todo.txt
ln todo.txt b.txt
ln -s todo.txt c.txt
mv todo.txt d.txt
rm b.txt
```

After rm:



In UNIX, Everything is a File

Directories are files of type “directory”

Additional types are:
“regular file”, “block device” (HDDs, SSDs), “pipe”, “socket” etc.

Directory inodes do not store pointers to data blocks
but rather filenames and pointers to inodes

What Data is Stored in an inode?

a Filename
b Containing Directory name
c File Size
d File type
e # of soft links to file
f location of soft links
g # of hard links to file
h location of hard links
i access rights
j timestamps
k file contents
l ordered list of data blocks

What Data is Stored in an inode? Solution

a Filename *No. Names are stored in directories*

b Containing Directory name *No. File can be in multiple dirs*

c File Size *Yes*

d File type *Yes*

e # of soft links to file *No (they are unknown)*

f location of soft links *No (they are unknown)*

g # of hard links to file *Yes (to know when to erase the file, check `stat`)*

h location of hard links *No (they are unknown to the inode)*

i access rights *Yes*

j timestamps *Yes*

k file contents *Sometimes*

l ordered list of data blocks *Yes, by definition*

Filesystem Caches Speed Up Writing to Disks

Writing data to the disk is slow, we can use a cache to speed it up

File blocks are cached in main memory in the *filesystem cache*
- Referenced blocks are likely to be referenced again (temporal locality)
- Logically near blocks are likely to be referenced (spatial locality)

A kernel thread (or daemon) writes changes periodically to disk

`flush` and `sync` system calls trigger a permanent write

Journaling Filesystem

Deleting a file on a Unix file system involves three steps:

1. Removing its directory entry.
2. Releasing the inode to the pool of free inodes.
3. Returning all disk blocks to the pool of free disk blocks.

Crashes could occur between any steps, leading to a storage leak

The journal contains operations in progress,
so if a crash occurs we can recover

inodes Are a Hybrid Allocaiton Strategy

inodes are offer greater flexibility over: contiguous, linked, FAT, or indexed

- Everything is a file on UNIX, names in a directory can be hard or soft links