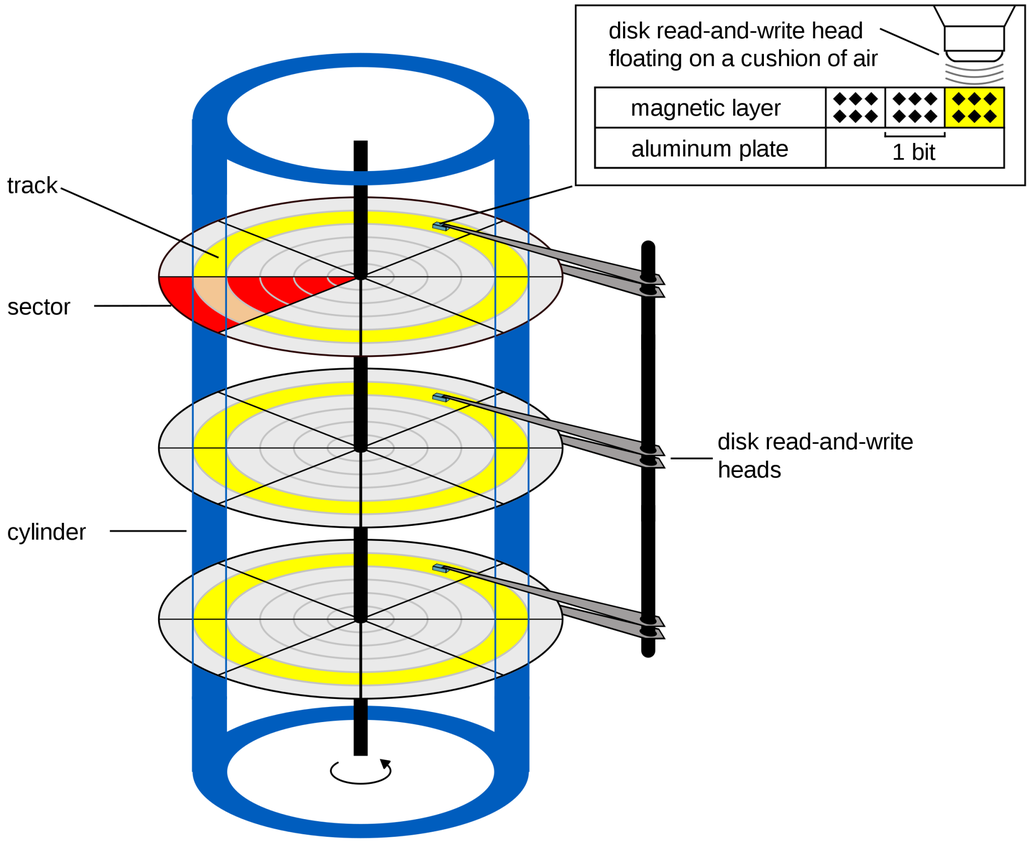
A Fast File System for UNIX

Exploiting spatial locality on disk.

Eugene Chou (euchou@ucsc.edu)

Hard Drive Schematic

- **Cylinder-Head-Sector (CHS)**
- **Sector:** a slice of a platter, typically 512 bytes.
- **Track:** concentric circle on a platter.
- **Cylinder:** a stack of tracks across platters.
- **Cylinder group:** one or more consecutive cylinders
- **Head:** device that performs the reads/writes.
  - Heads connected by an **arm**.



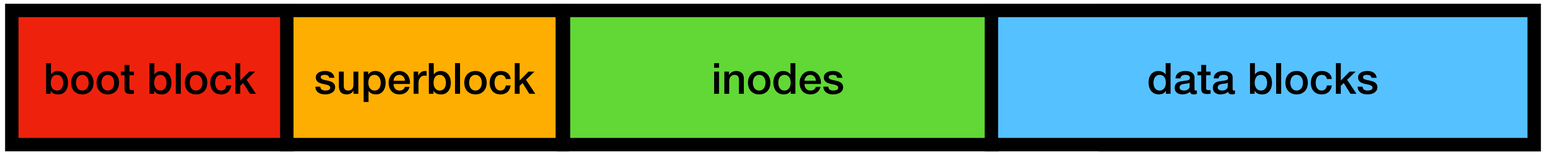
Unix File System (UFS)

- “Simple” programmer interface.
- Reads/writes 512-bytes at a time.
- Used on PDP-11 and VAX-11.
- Terrible throughput.
  - Around 2% of maximum disk bandwidth.

- By Stefan_Kögl - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=466937
- By Emiliano Russo, Associazione Culturale VerdeBinario - Public Domain, https://commons.wikimedia.org/w/index.php?curid=4807141

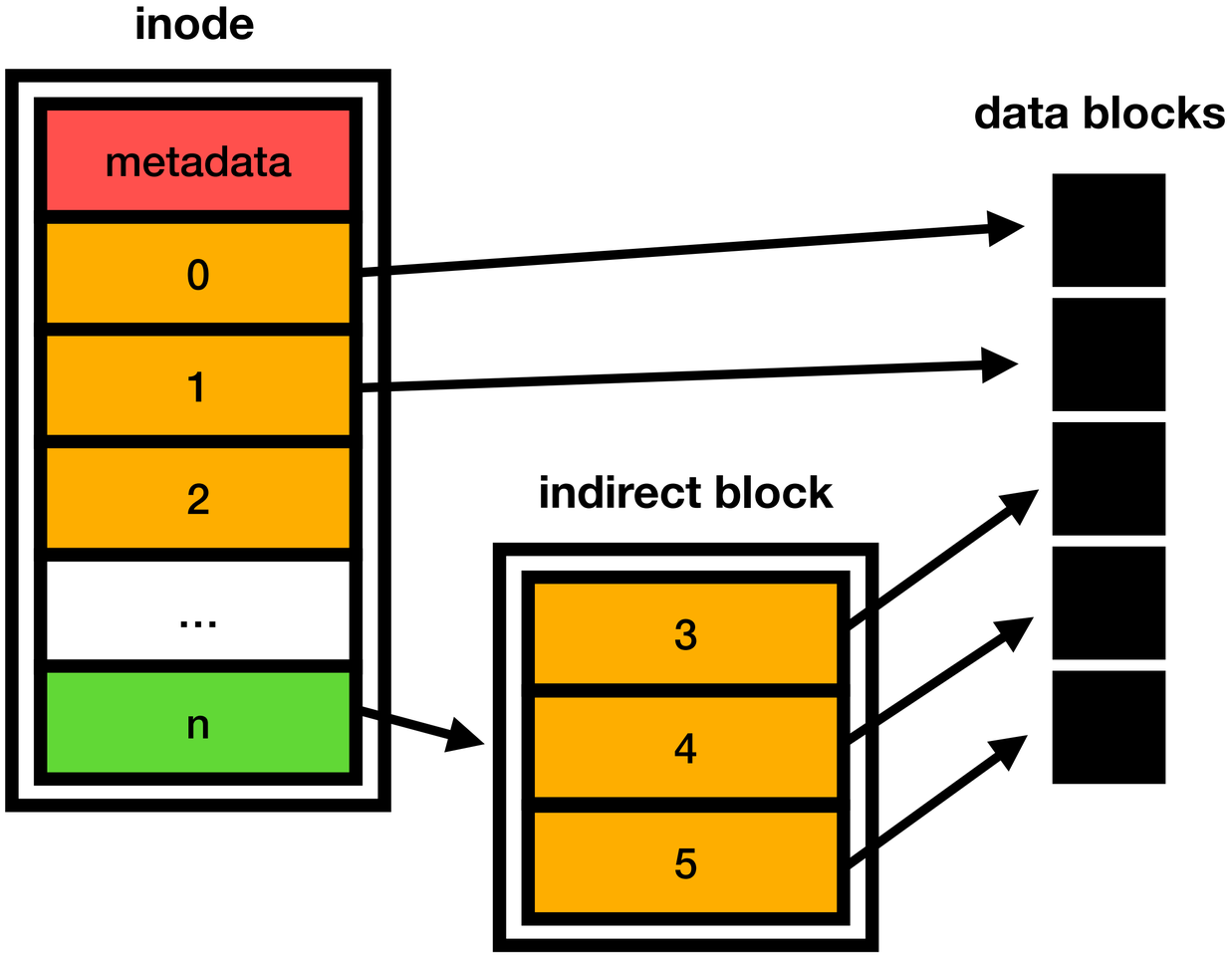
UFS Layout

- Free data blocks tracked with a **free list**.
  - Linked list of free data blocks.
  - Pointer to list in superblock.



Inodes

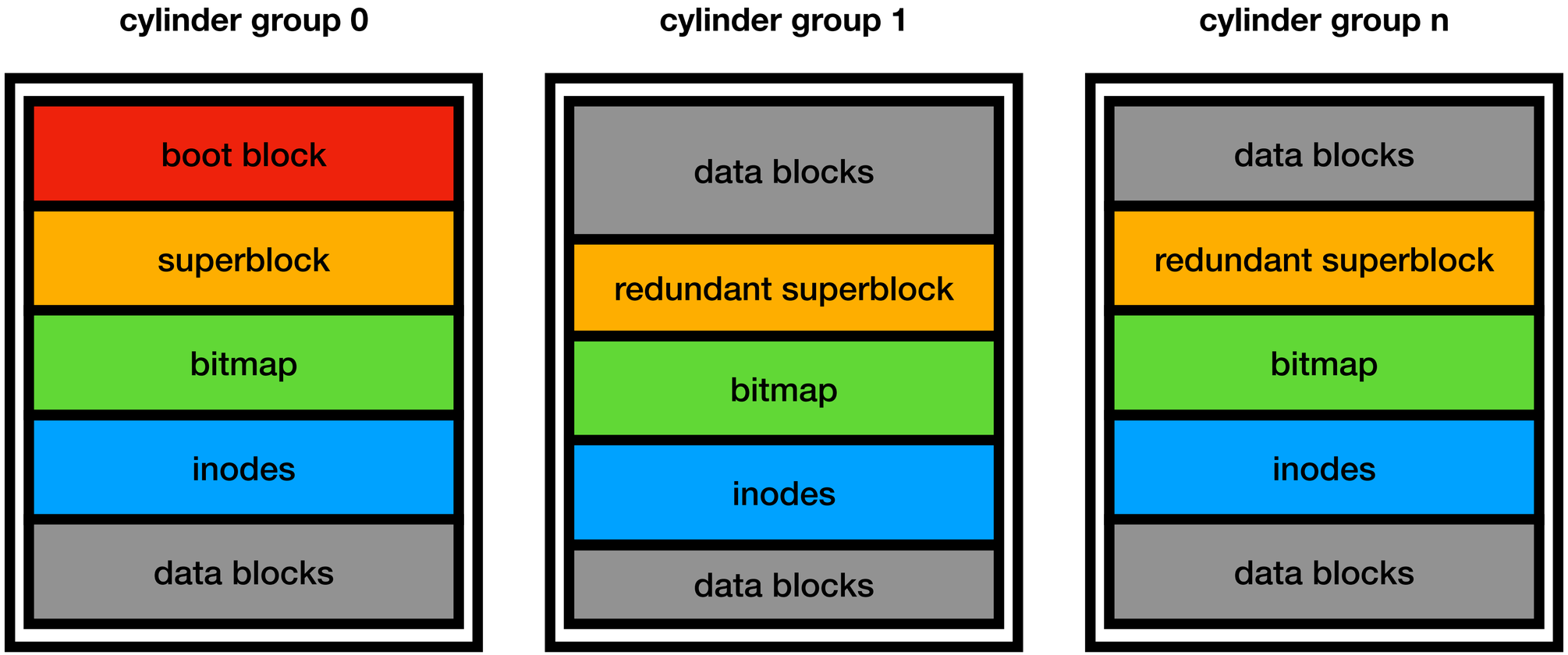
- Describes files.
  - Everything in UNIX is a file.
- Identified with an *i-number*.
- Files made up of data blocks.
- Indirect blocks can point at indirect blocks.



Goals for Fast File System (FFS)

- Better locality for inodes and data blocks.
- Faster throughput for small *and* large files.
- Flexible for different processor/storage characteristics.
- Enhance programmer interface.

FFS Disk Layout



Redrawn based on: https://docs.oracle.com/cd/E19455-01/805-7228/6j6q7uf0r/index.html

Bigger blocks

- Good for large files.
  - More data transfer per disk transaction.
- Bad for small files.
  - File systems are typically made up of small files.
- In general, this means more wasted space.

Fragmenting Blocks

- Break blocks into 2, 4, or 8 addressable fragments.
- Smallest fragment is the size of a sector.
- Store big files using as many blocks as possible.
- Remaining data goes into fragments.
- Smaller files can use available fragments!

Knowledge is Power

- UFS doesn’t account for underlying hardware.
- FFS stores more information to make better decisions.
  - How fast does the disk spin?
  - How far apart to place blocks for a single file?
- FFS parameterizes processor/storage properties to make this possible.
  - Even provides optimal block size for specific applications.

Placing Directories and Files

- FFS layout policies split into global and local policies.
- Global policies:
  - How do we cluster inodes and data blocks?
  - Should we seek to another cylinder group?
- Local policies:
  - How should data blocks be laid out?

How Much Faster For Reads?

- Tests run on VAX-11/750.
- No data processing by any test programs.
- Programs run *at least three times* in succession.
- File system had 10% free space reserve.
  - Halved performance with full file system.

Filesystem Type	Processor and bus measured	Speed (KBytes/s)	Read bandwidth (%)	% CPU
Old 1024	750/UNIBUS	29	3	11
New 4096/1024	750/UNIBUS	221	22	43
New 8192/1024	750/UNIBUS	233	24	29
New 4096/1024	750/MASSBUS	466	47	73
New 8192/1024	750/MASSBUS	466	47	54

Data from McKusick, et al. *A Fast File System for UNIX*

How Much Faster For Writes?

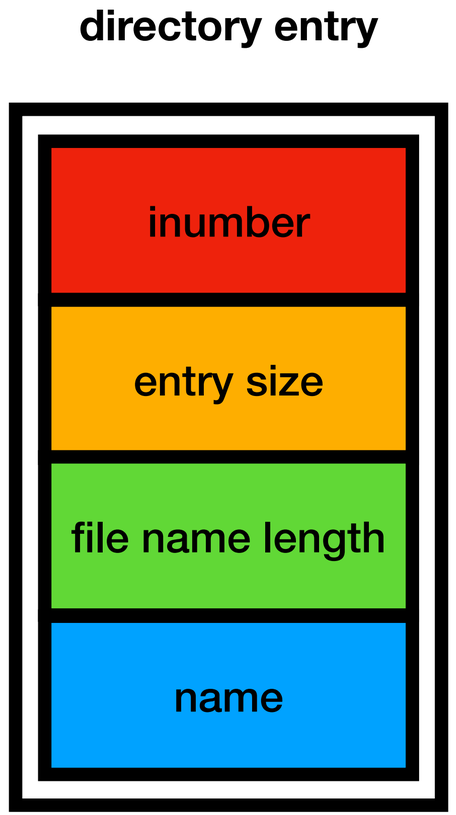
- Tests run on VAX-11/750.
- No data processing by any test programs.
- Programs run *at least three times* in succession.
- File system had 10% free space reserve.
  - Halved performance with full file system.

Filesystem Type	Processor and bus measured	Speed (KBytes/s)	Write bandwidth (%)	% CPU
Old 1024	750/UNIBUS	48	3	29
New 4096/1024	750/UNIBUS	142	14	43
New 8192/1024	750/UNIBUS	215	22	46
New 4096/1024	750/MASSBUS	323	33	94
New 8192/1024	750/MASSBUS	466	47	95

Data from McKusick, et al. *A Fast File System for UNIX*

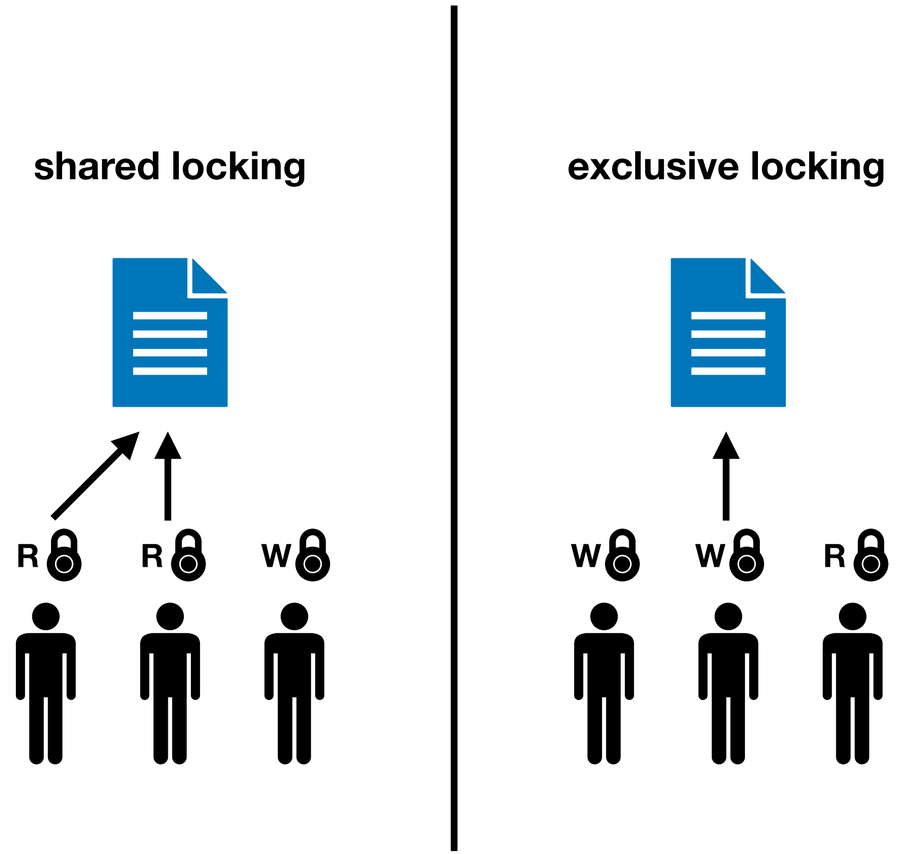
Longer File Names

- Maximum file name length: 255
  - But they claim this is “nearly arbitrary length.”
- Directories allocated in 512-byte chunks.
  - Each chunk contains an *entry*.



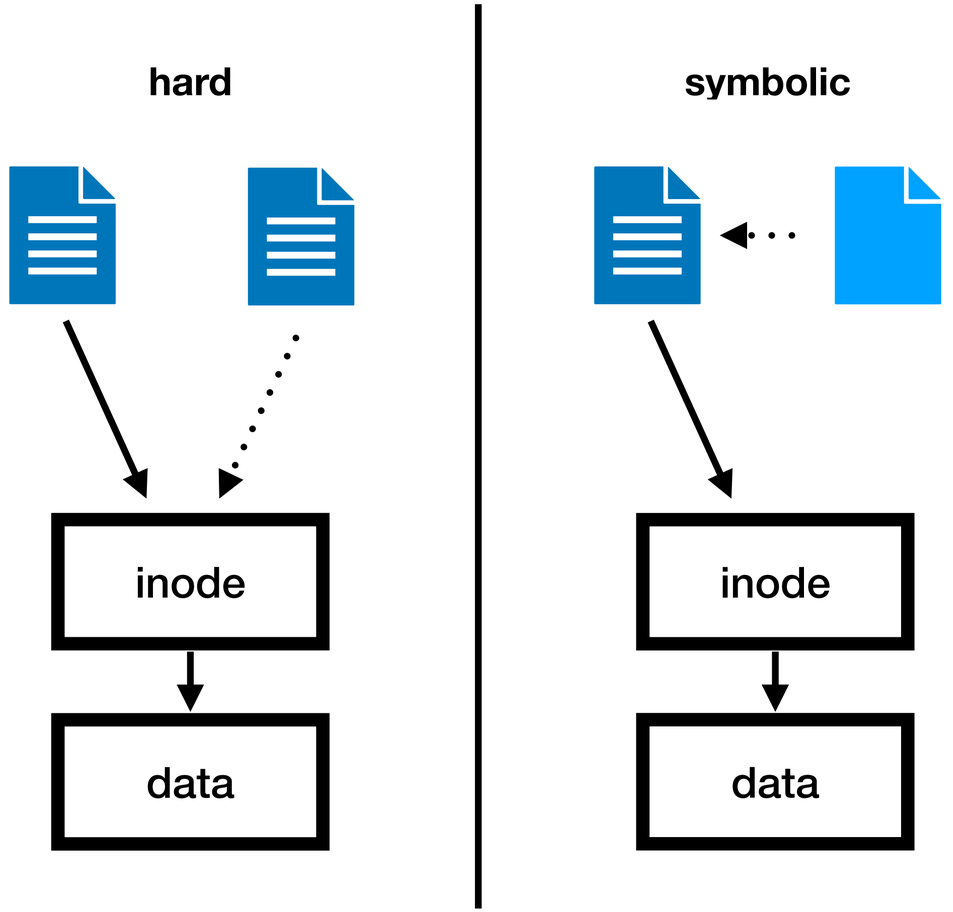
Locking Files

- Files are locked for concurrent updates.
- Two schemes:
  - Hard locking
  - Advisory locking ⇒ used in FFS.
- Advisory locking uses **shared** and **exclusive** locks.



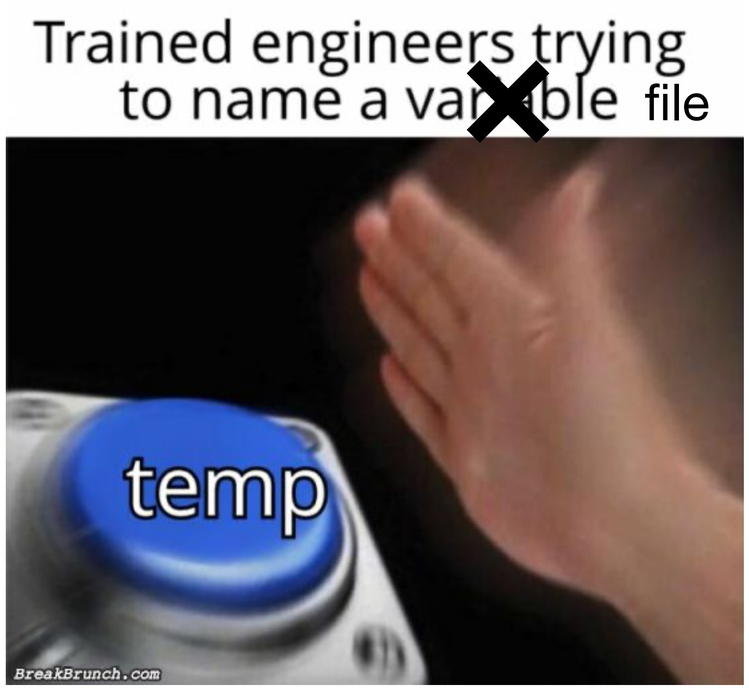
Symbolic Links

- Each file is linked to an inode.
  - Files can appear in multiple directories.
- UFS only supported **hard links**.
  - Can’t be used across file systems.
- Symbolic links are just files with a pathname.
  - Link count *not* incremented.



Renaming

- Programs that renamed required a temporary file.
- UFS required three system calls to rename.
- Failure with system or program ⇒ file isn’t moved properly.
  - May end up with temporary name instead.



Protecting Users From Other Users

- Users could originally allocate all available resources.
- **Quotas** are set per user to enforce limits.
  - Capped number of inodes.
  - Capped number of data blocks.
- Users reprimanded if they go over quota.

A Couple Questions

- How would flash memory affect this file system design?
- Were the performed tests rigorous?
- Why didn’t they attempt deadlock detection with their advisory file locking?