

#### of NORTH CAROLINA

#### What the VFS does

COMP 790: OS Implementation

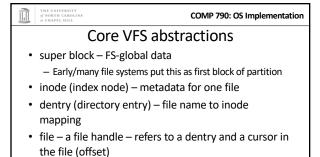
- The VFS is a substantial piece of code, not just an API wrapper
- Caches file system metadata (e.g., file names, attributes)
  - Coordinates data caching with the page cache
- Enforces a common access control model
- Implements complex, common routines, such as path lookup, file opening, and file handle management

#### 7

## COMP 790: OS Implementation FS Developer's Perspective FS developer responsible for implementing a set of standard objects/functions, which are called by the VFS Primarily populating in-memory objects from stable storage, and writing them back Can use block device interfaces to schedule disk I/O And page cache functions And some VFS helpers Analogous to implementing Java abstract classes

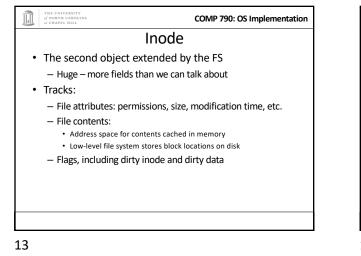
COMP 790: OS Implementation High-level FS dev. tasks • Translate between volatile VFS objects and backing storage (whether device, remote system, or other/none) – Potentially includes requesting I/O • Read and write file pages

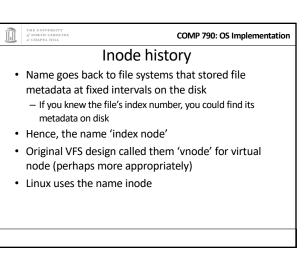
9



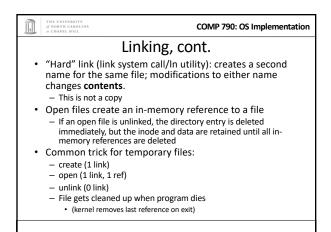
10

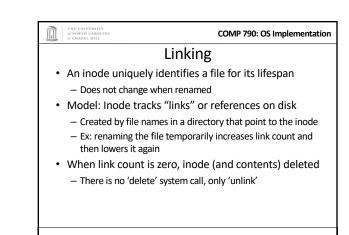
## COMP 790: OS Implementation Super blocks • SB + inodes are *extended* by FS developer • Stores all FS-global data – Opaque pointer (s\_fs\_info) for fs-specific data • Includes many hooks for tasks such as creating or destroying inodes • Dirty flag for when it needs to be synced with disk • Kernel keeps a circular list of all of these





THE UNIVERSITY of NORTH CAROLINA af CHAPEL HILL	COMP 790: OS Implementation	
Embedded inodes		
<ul> <li>Many file systems embed t FS-specific inode, e.g.,:</li> </ul>	he VFS inode in a larger,	
<pre>struct donfs_inode {</pre>		
int ondisk_blocks[];		
/* other stuff*/		
struct inode vfs_inode;	;	
}		
Why? Finding the low-leve inode just requires simple (		
15		



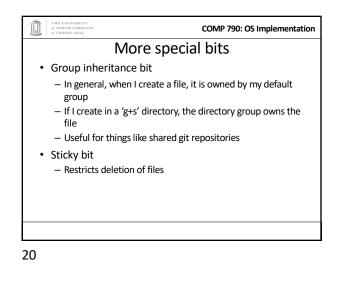


	THE UNIVERSITY of NORTH CAROLINA af CHAPEL HILL	COMP 790: OS Implementation	
Inode 'stats'			
• -	The 'stat' word encodes both per	missions and type	
<ul> <li>High bits encode the type: regular file, directory, pipe, char device, socket, block device, etc.</li> <li>Unix: Everything's a file! VFS involved even with sockets!</li> </ul>			
<ul> <li>Lower bits encode permissions:         <ul> <li>3 bits for each of User, Group, Other + 3 special bits</li> <li>Bits: 2 = read, 1 = write, 0 = execute</li> <li>Ex: 750 – User RWX, Group RX, Other nothing</li> </ul> </li> </ul>			



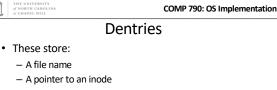
#### Special bits

- For directories, 'Execute' means search
  - X-only permissions means I can find readable
  - subdirectories or files, but can't enumerate the contents
     Useful for sharing files in your home directory, without sharing your home directory contents
  - Lots of information in meta-data!
- Setuid bit
  - Mostly relevant for executables: Allows anyone who runs this program to execute with owner's uid
  - Crude form of permission delegation
- 19



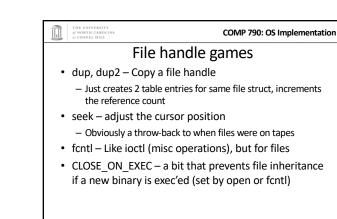
COMP 790: OS Implementation File objects • Represent an open file; point to a dentry and cursor – Each process has a table of pointers to them – The int fd returned by open is an offset into this table • These are VFS-only abstractions; the FS doesn't need to track which process has a reference to a file • Files have a reference count. Why? – Fork also copies the file handles – If your child reads from the handle, it advances your (shared) cursor

21



- A parent pointer (null for root of file system)
- Ex: /home/porter/vfs.pptx would have 4 dentries:
  - /, home, porter, & vfs.pptx
  - Parent pointer distinguishes /home/porter from /tmp/porter
- These are also VFS-only abstractions

   Although inode hooks on directories can populate them



22

### COMP 790: OS Implementation Why dentries? • A simple directory model might just treat it as a file listing <name, inode> tuples • Why not just use the page cache for this? • FS directory tree traversal very common; optimize with special data structures • The dentry cache is a complex data structure we will discuss in much more detail later

#### of NORTH CAROLINA at CHAPEL HILL

COMP 790: OS Implementation

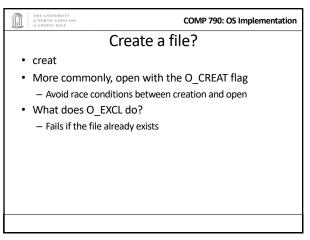
### Summary of abstractions

- Super blocks FS- global data
- Inodes stores a given file
- File (handle) Essentially a <dentry, offset> tuple
- Dentry Essentially a <name, parent dentry, inode> tuple

# COMP 790: OS Implementation More on the user's perspective • Let's wrap today by discussing some common FS system calls in more detail • Let's play it as a trivia game – What call would you use to...

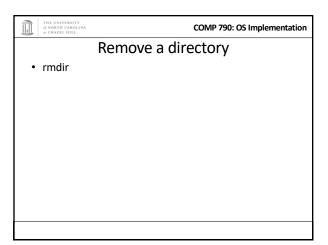
26

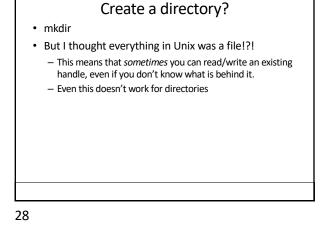
THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL



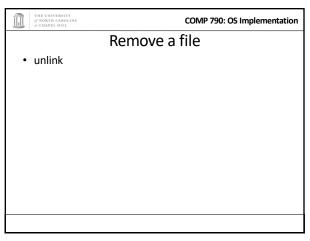
27

25

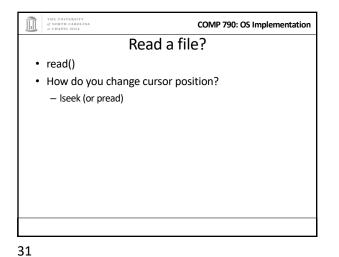


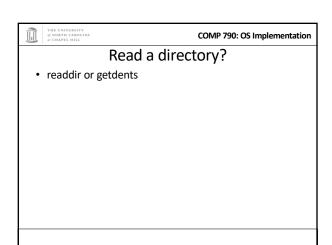


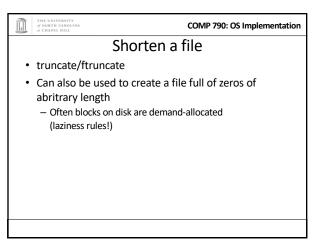
COMP 790: OS Implementation

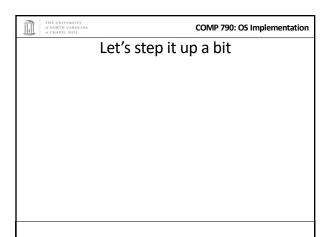


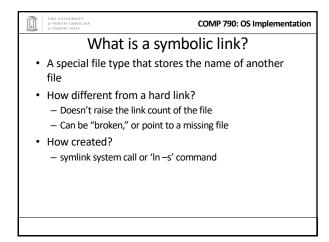


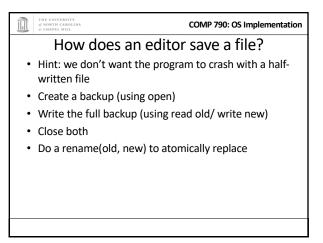


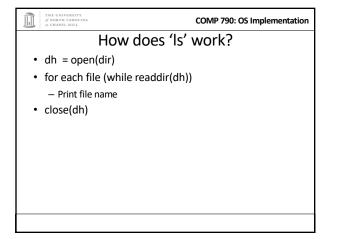












	THE UNIVERSITY of NORTH CAROLINA af CHAPEL HILL	COMP 790: OS Implementation		
	Summary			
•	Today's goal: VFS overview fro – User (application programmer) – FS implementer • Used many page cache and disk I/ Key VFS objects Important to be able to pick P from a line up	O tools we've seen		
	<ul> <li>Home and ap</li> <li>Homework: think about pseudo command-line file system utiliti</li> </ul>			

