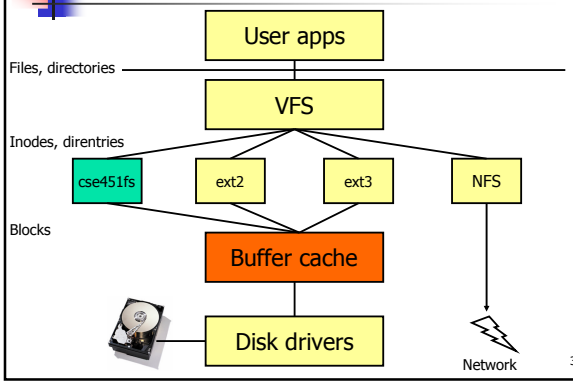
Section 9:

Project 3 – The Buffer Cache
Network File Systems

Questions?

Linux FS Layers (Revisit)



Linux Buffer Manager

- Buffer cache caches disk blocks & buffers writes
  - When you write to a file, the data goes into buffer cache (for write-back buffer caches)
  - Sync is required to flush the data to disk
    - Update and bdflush processes flush data to disk (every 30s)
- Linux buffer cache code fundamentals:
  - Blocks are represented by *buffer_heads*
    - Actual data is in *buffer_head->b_data*
  - For a given disk block, buffer manager could be:
    - Completely unaware of it
      - no *buffer_head* exists, block not in memory
    - Aware of block information
      - *buffer_head* exists, but block data (*b_data*) not in memory
    - Aware of block information and data
      - Both the *buffer_head* and its *b_data* are valid ("$ hit")

Accessing blocks

- To read a block, FS uses sb_bread(…):
  - Find the corresponding *buffer_head*
    - Create if doesn't exist
  - Make sure *buffer_head->b_data* is in memory (read from disk if necessary)
- To write a block:
  - mark_buffer_dirty() + brelse() - mark buffer as changed and release to kernel (which does the writing)

Some buffer manager functions

cse451_bread(pbh, inode, block, create)	Get the *buffer_head* for the given disk block, ensuring that the data is in memory and ready for use. Increments ref count; always pair with a brelse.
cse451_getblk(pbh, inode, block, create)	Get the *buffer_head* for the given disk block. Does not guarantee anything about the state of the actual data. Increments ref count; always pair with a brelse. Zeros out new blocks (required for security).
brelse(bh)	Decrement the ref. count of the given buffer.
mark_buffer_dirty(bh)	Mark the buffer modified, meaning needs to be written to disk at some point.
mark_buffer_uptodate(bh)	Indicate that the data pointed to by bh is valid.

[Remember this lock-release pattern for future use in multi-threaded (multi-process) programs; it's how reference-counted pointers also work.]

Network File Systems

- Provide access to remote files over a network
  - Typically aim for location and network transparency
- Designed and optimized for different types of operations
  - E.g., work over LANs, over WANs, support disconnected operations, fault-tolerance, scalability, consistency, etc.

Examples:

- Network File System (NFS) – Sun Microsystems
- Server Message Block (SMB) – originally IBM, Microsoft
- Andrew File System (AFS) – CMU
- Coda – CMU

NFS

- A server exports (or shares) a directory
- A client mounts the remote directory onto his local FS namespace
  - The mounted directory looks like an integral subtree of the local file system, replacing the subtree descending from the local directory [1]
  - However, it's all namespace magic, nothing is actually stored on local disks

[1] *http://www.csie.fju.edu.tw/~yeh/courses/spring08/os/ch11.ppt*

Mounting an NFS Export

- A remote exported directory can be "glued" onto the local hierarchy

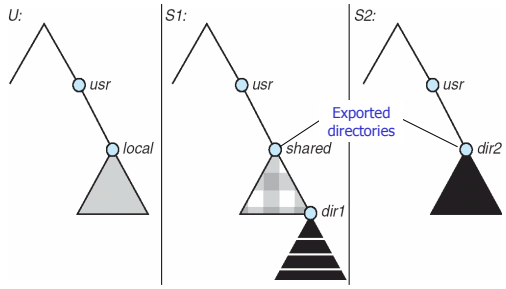


Figure taken from *http://www.csie.fju.edu.tw/~yeh/courses/spring08/os/ch11.ppt*

Mounting NFS Directories

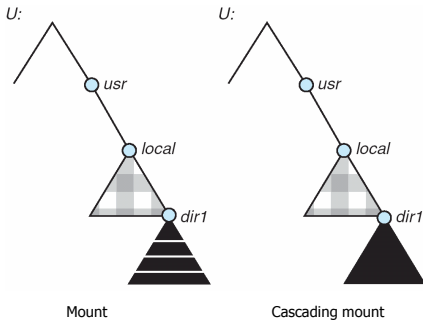


Figure taken from *http://www.csie.fju.edu.tw/~yeh/courses/spring08/os/ch11.ppt*

The NFS Protocol

- NFS is designed to operate in a heterogeneous environment of different machines, operating systems, and network architectures
  - NFS specifications are independent of these media
- This independence is achieved through the use of RPC and XDR (eXternal Data Representation)
- Nearly one-to-one correspondence between regular UNIX file system calls and the NFS protocol RPCs
  - looking up a file within a directory
  - reading a set of directory entries
  - accessing file attributes
  - reading and writing files

Bullets taken from presentation at *http://www.csie.fju.edu.tw/~yeh/courses/spring08/os/ch11.ppt*

The NFS Architecture

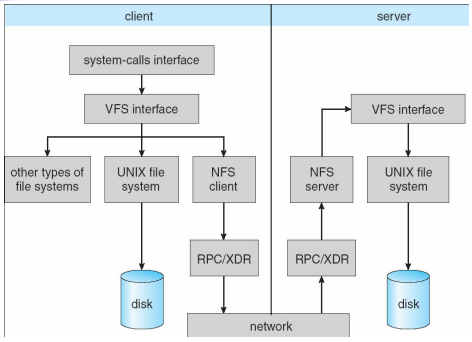


Figure taken from *http://www.csie.fju.edu.tw/~yeh/courses/spring08/os/ch11.ppt*

Example: Setting up an NFS Share

- Server exports directory
  - Check nfsd is running and it if not (e.g., service nfsd start)
  - Edit /etc/exports file
    - `/usr/shared  192.168.0.0/255.255.255.0(rw)`
    - `man exports` for more detailed structure of file
  - Force nfsd to re-read /etc/exports using `exportfs -ra`
- Client mounts the remote directory locally
  - `mount -t nfs 192.168.0.1:/usr/share /usr/local`

    (192.168.0.1 is the server's IP address)
  - can enable automatic mounting by editing `/etc/fstab` (`man fstab`)

Note:
- The above is just a "by-hand" example; use the Internet for more precise tutorials and troubleshooting

13