17. Journaling & Mounting

CS 4352 Operating Systems

Adopted from slides of Drs. Silberschatz, Huang, & Bryant

Journaling

- A file system check can be very slow, especially for large file systems
- An alternative is journaling, also called write-ahead logging
- Simple idea: before updating the on-disk structures, write information about the update to a journal (also called a log)
    - If there's a crash before the disk can be updated, the journal contains enough information to recover
        - You don't have to scan the entire disk!

Example: ext2 vs ext3

- The basic layout of ext2 (no journaling):

- The basic layout of ext3 (with journaling):

Journaling Phases

- Four basic phases:
  - Journal write: write contents of transaction to journal; wait for these to complete

  - Journal commit: write the transaction commit block; wait for write to complete; transaction is committed

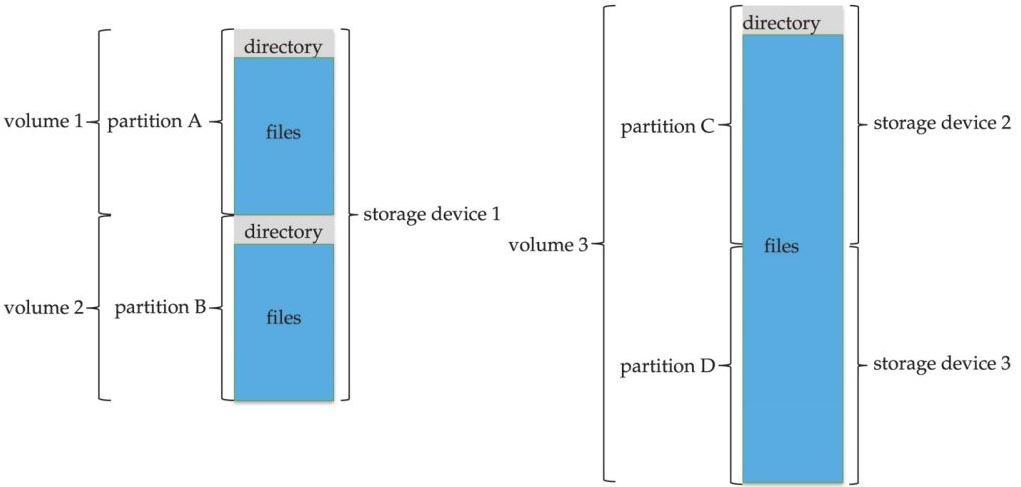
  - Checkpoint: write the contents of the update (metadata and data) to disk
  - Free: mark the transaction as “free” in the journal by updating the journal superblock
    - Done at “some point” in the future

Extents

- Recall that ext2 and ext3 use indirect pointers to data blocks
  - Can be inefficient!
- The ext4 file system uses extents
  - An extent specifies an (1) an initial block address, and (2) the number of blocks in the extent
- Extents exercise
  - Follow the steps in this link to understand ext4 extents in more depth
    - https://digital-forensics.sans.org/blog/2010/12/20/digital-forensics-understanding-ext4-part-1-extents
  - You'll need a hex-editor
    - Emacs includes one
      - emacs <filename>, hold down <Alt>-x, release it, then type hexl-mode
    - Vim needs to use xxd to convert file
      - vim <filename>, run :%!xxd

Storage Device Organization Review

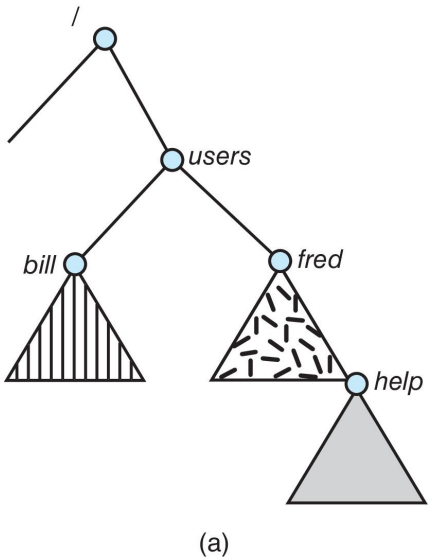
- General-purpose computers can have multiple storage devices
  - Devices can be sliced into partitions, which hold volumes
  - Volumes can span multiple partitions
  - Each volume usually formatted into a file system
  - # of file systems varies, typically dozens available to choose from
- A typical organization



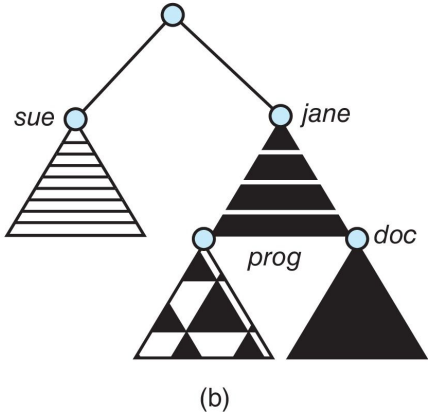
Partitions and Mounting

- Partition can be a volume containing a file system (“cooked”) or raw – just a sequence of blocks with no file system
- Boot block can point to boot volume or boot loader set of blocks that contain enough code to know how to load the kernel from the file system
- Root partition contains the OS, other partitions can hold other OSes, other file systems, or be raw
  - Mounted at boot time
  - Other partitions can mount automatically or manually on mount points – location at which they can be accessed
- At mount time, file system consistency is checked
  - Is all metadata correct?
    - If not, fix it, try again
    - If yes, add to mount table, allow access

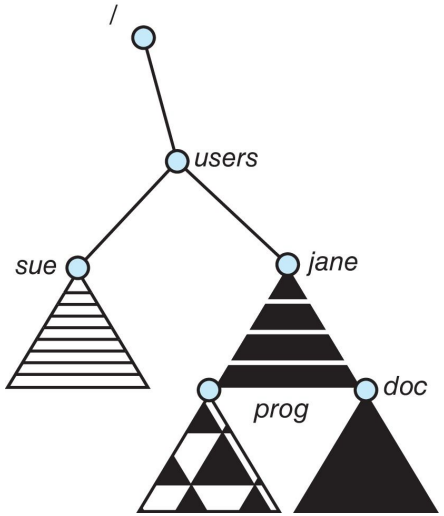
File Systems and Mounting



Existing system



Unmounted file system



After mounting (b) into the existing directory tree

Example Mount Points and File Systems - Solaris

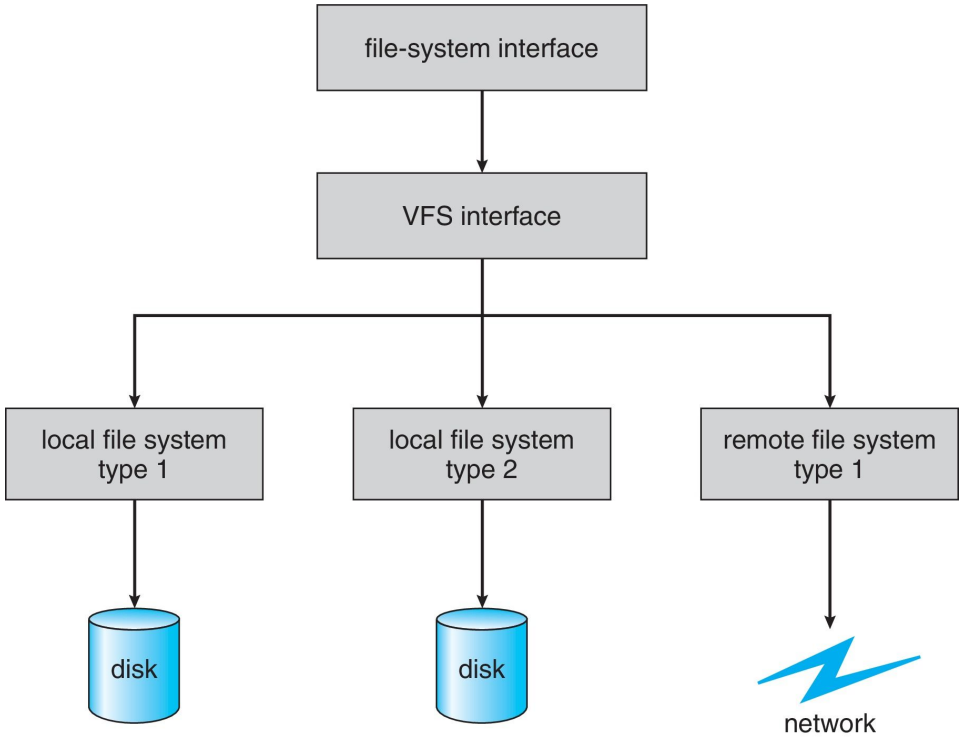
/	ufs
/devices	devfs
/dev	dev
/system/contract	ctfs
/proc	proc
/etc/mnttab	mntfs
/etc/svc/volatile	tmpfs
/system/object	objfs
/lib/libc.so.1	lofs
/dev/fd	fd
/var	ufs
/tmp	tmpfs
/var/run	tmpfs
/opt	ufs
/zpbge	zfs
/zpbge/backup	zfs
/export/home	zfs
/var/mail	zfs
/var/spool/mqueue	zfs
/zpbg	zfs
/zpbg/zones	zfs

Virtual File Systems

- Virtual File Systems (VFS) on Unix provide an object-oriented way of implementing file systems
- VFS allows the same system call interface (the API) to be used for different types of file systems
  - Separates file-system generic operations from implementation details
  - Implementation can be one of many file systems types, or network file system
    - Implements vnodes which hold inodes or network file details
  - Dispatches operation to appropriate file system implementation routines

Virtual File Systems (Cont.)

- The API is to the VFS interface, rather than any specific type of file system



Sharing of Files across a Network

- First method involved manually sharing each file – programs like ftp
- Second method uses a distributed file system (DFS)
  - Remote directories visible from local machine
- Third method – World Wide Web
  - A bit of a revision to first method
  - Use browser to locate file/files and download /upload
  - Anonymous access doesn’t require authentication

The Sun Network File System (NFS)

- An implementation and a specification of a software system for accessing remote files across LANs (or WANs)
- The implementation originally part of SunOS operating system, now industry standard/very common
- Can use unreliable datagram protocol (UDP/IP) or TCP/IP, over Ethernet or other network
- NFS is designed to operate in a heterogeneous environment of different machines, operating systems, and network architectures; the NFS specifications independent of these media

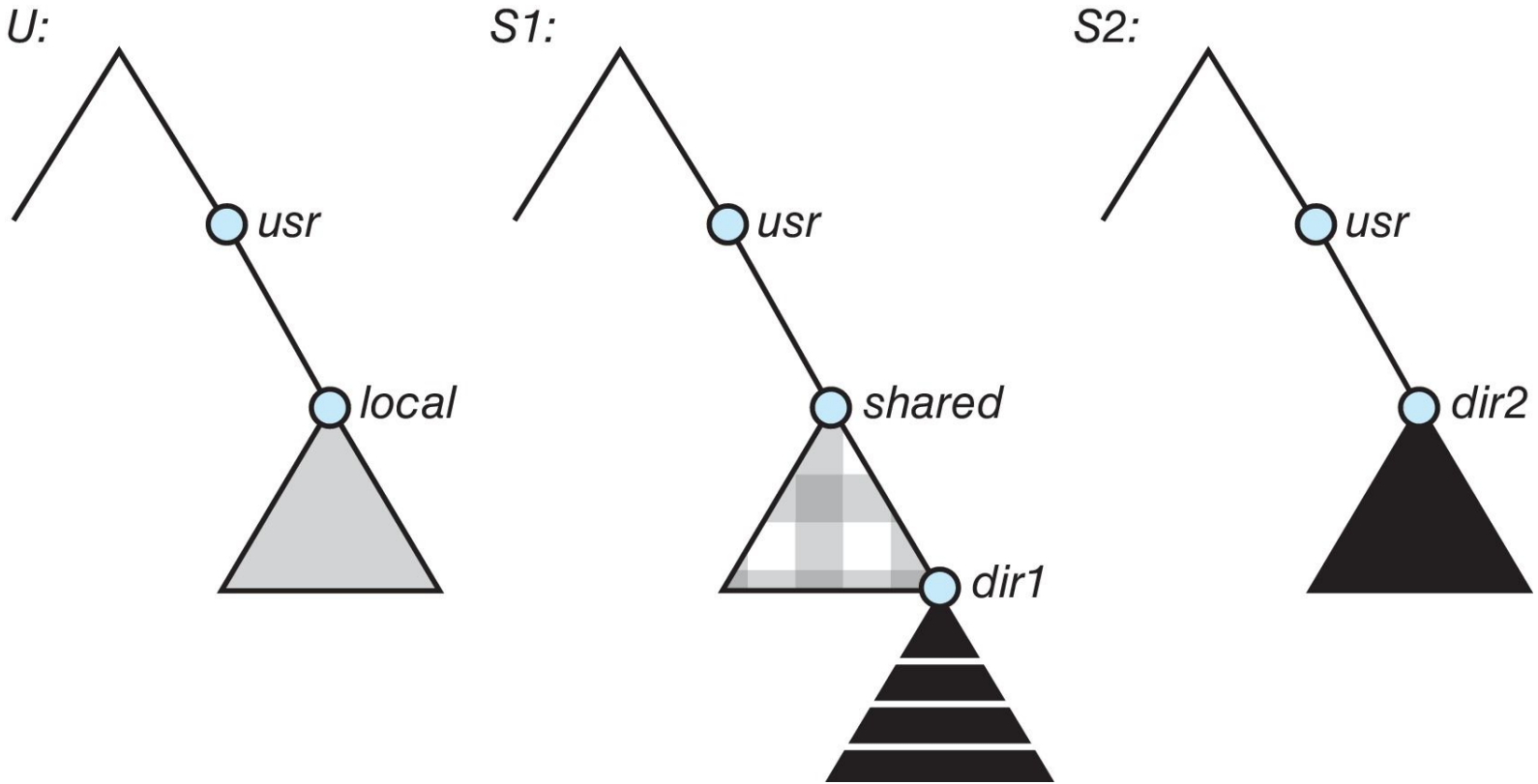
NFS (Cont.)

- Interconnected workstations viewed as a set of independent machines with independent file systems, which allows sharing among these file systems in a transparent manner
  - A remote directory is mounted over a local file system directory
    - The mounted directory looks like an integral subtree of the local file system, replacing the subtree descending from the local directory
  - Specification of the remote directory for the mount operation is non-transparent; the host name of the remote directory has to be provided
    - Files in the remote directory can then be accessed in a transparent manner
  - Subject to access-rights accreditation, potentially any file system (or directory within a file system), can be mounted remotely on top of any local directory

NFS Mount Protocol

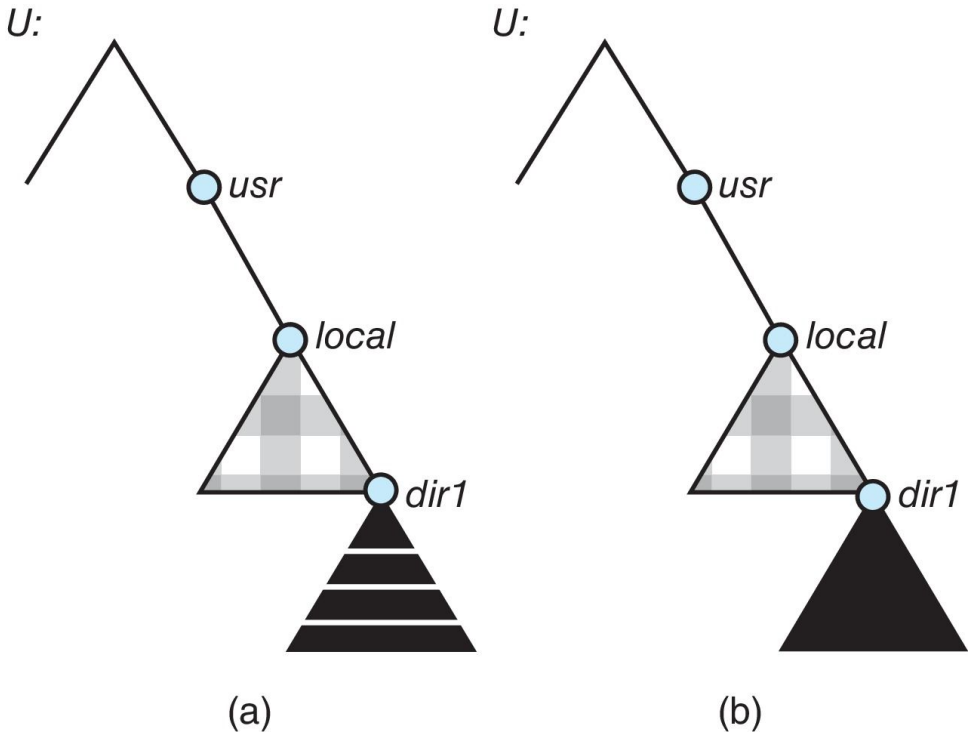
- Establishes initial logical connection between server and client
- Mount operation includes name of remote directory to be mounted and name of server machine storing it
    - Mount request is mapped to corresponding RPC and forwarded to mount server running on server machine
    - Export list – specifies local file systems that server exports for mounting, along with names of machines that are permitted to mount them
- Following a mount request that conforms to its export list, the server returns a file handle—a key for further accesses
    - File handle – a file-system identifier, and an inode number to identify the mounted directory within the exported file system

Three Independent File Systems



Mounting in NFS

- (a) shows the effects of mounting S1:/usr/shared over U:/usr/local
  - The original directory /usr/local on that machine is no longer visible
- (b) shows cascading mounting S2:/usr/dir2 over U:/usr/local/dir1
  - Users can access files within dir2 on U using the prefix /usr/local/dir1



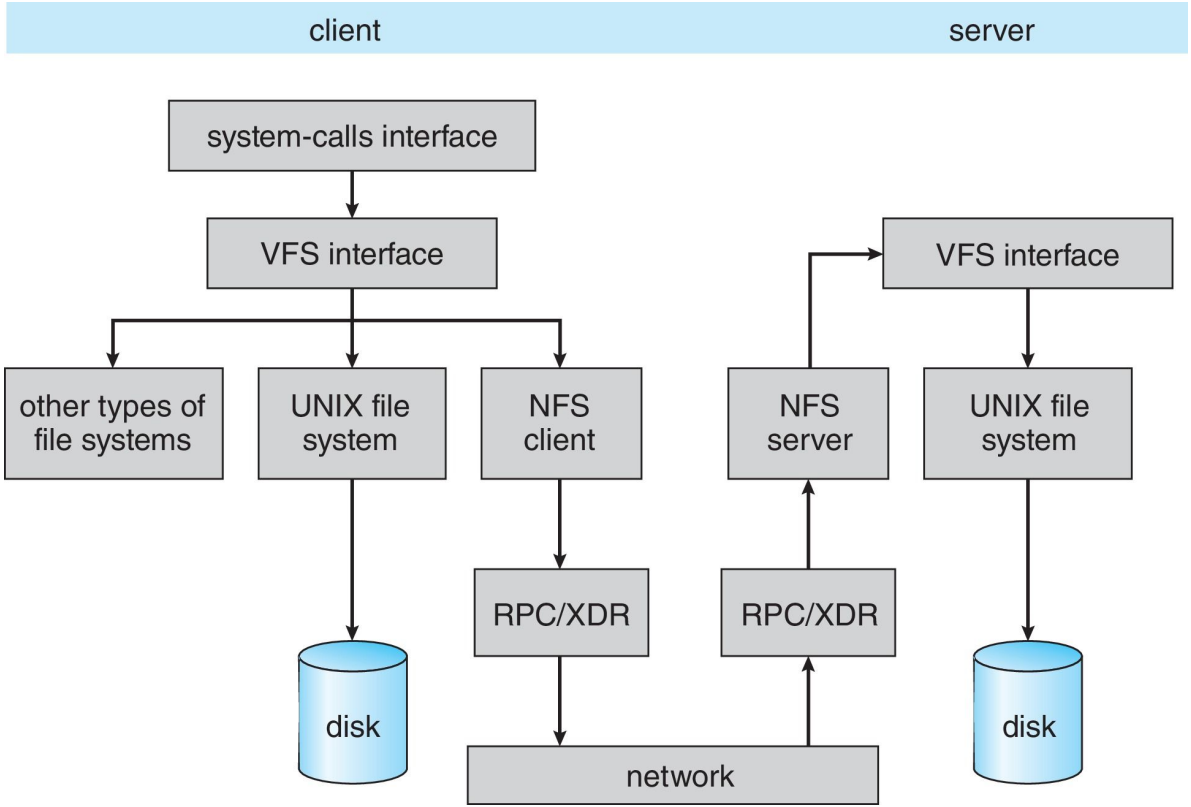
NFS Protocol

- Provides a set of remote procedure calls (RPCs) for remote file operations
  - searching for a file within a directory
  - reading a set of directory entries
  - manipulating links and directories
  - accessing file attributes
  - reading and writing files
- NFS servers are stateless; each request has to provide a full set of arguments (NFS V4 is newer, less used – very different, stateful)
- The NFS protocol does not provide concurrency-control mechanisms

Three Major Layers of NFS Architecture

- UNIX file-system interface (based on the open, read, write, and close system calls, and file descriptors)
- Virtual File System (VFS) layer – distinguishes local files from remote ones, and local files are further distinguished according to their file-system types
  - The VFS activates file-system-specific operations to handle local requests according to their file-system types
  - Calls the NFS protocol procedures for remote requests
- NFS service layer – bottom layer of the architecture
  - Implements the NFS protocol

Schematic View of NFS Architecture



Homework

- Chapter 16
- Read “smashing the stack for fun and profit”
  - Already on BB

Next Lecture

We start looking at some security problems