Operating Systems
[15. File-System Internals]

Chung-Wei Lin

cwlin@csie.ntu.edu.tw

CSIE Department

National Taiwan University

From Operating System Concepts (10th Edition)

Objectives

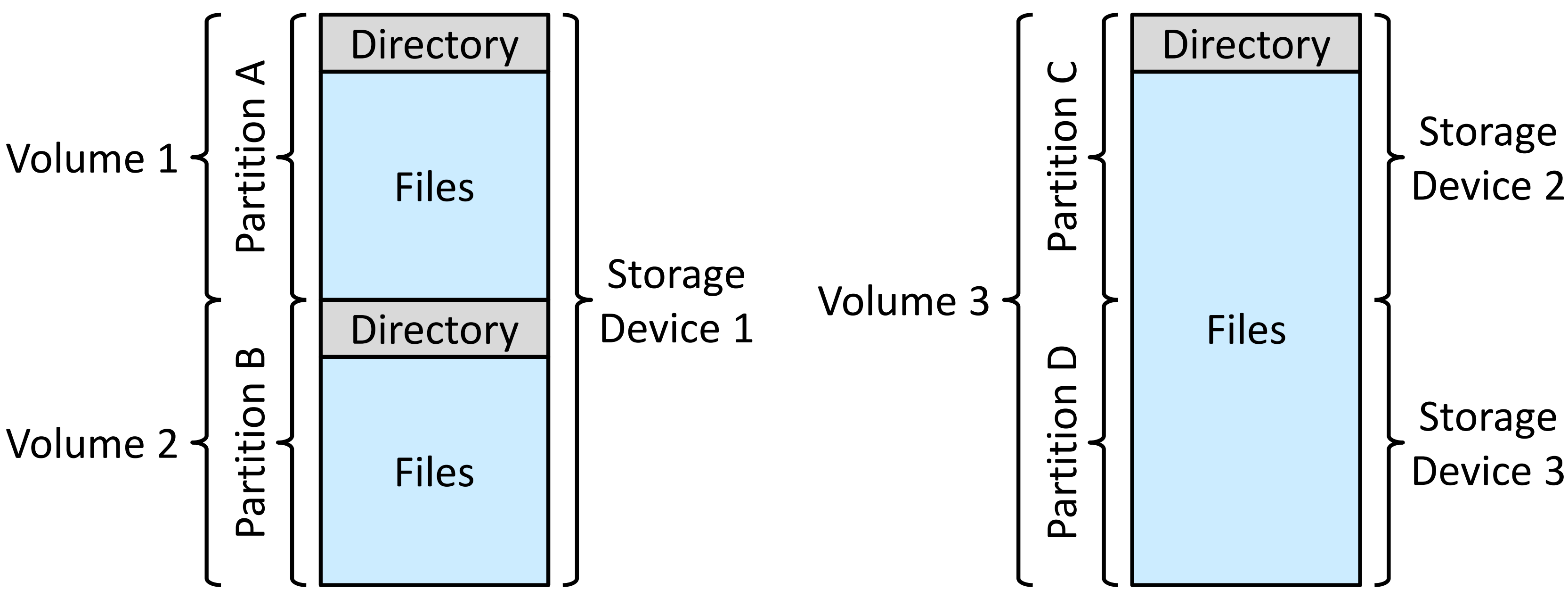
- Delve into the details of file systems and their implementation
- Explore booting and file sharing
- Describe remote file systems, using NFS as an example

Outline

- **File Systems**
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- Consistency Semantics
- NFS

File Systems (1/2)

- A general-purpose computer system can have multiple storage devices
- A device can be sliced up into partitions, which hold volumes, which in turn hold file systems
  - Depending on the volume manager, a volume may span multiple partitions as well



File Systems (2/2)

- Computer systems may also have varying numbers of file systems, and the file systems may be of varying types
  - Example: Solaris
    - **tmpfs**: a "temporary" file system that is created in volatile main memory
    - **objfs**: a "virtual" file system that gives debuggers access to kernel symbols
    - **ctfs**: a virtual file system that maintains "contract" information to manage which processes start when booting and must continue during operation
    - **lofs**: a "loop back" file system that allows one file system to be accessed in place of another one
    - **procfs**: a virtual file system that presents information on all processes
    - **ufs**, **zfs**: general-purpose file systems

/	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

Outline

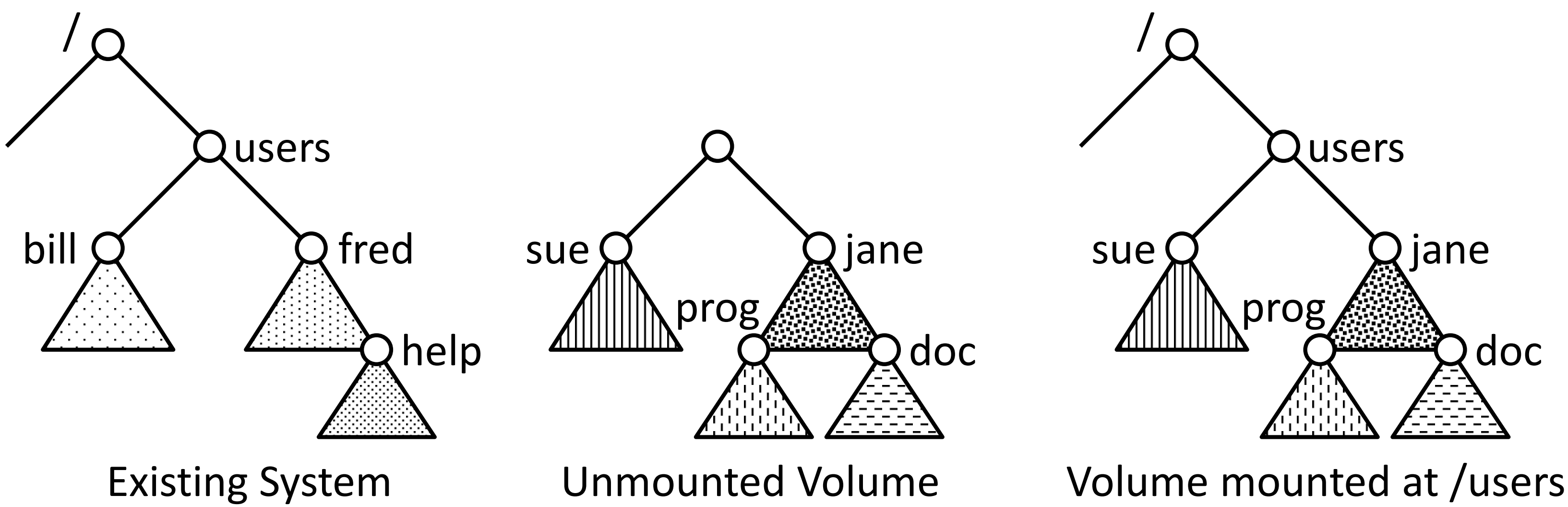
- ❑ File Systems
- ❑ **File-System Mounting**
- ❑ Partitions and Mounting
- ❑ File Sharing
- ❑ Virtual File Systems
- ❑ Remote File Systems
- ❑ Consistency Semantics
- ❑ NFS

File-System Mounting (1/2)

- A file system must be mounted before it can be available to processes on the system
  - The OS is given the device name and the **mount point**, the location within the file structure where the file system is to be attached
    - Some require that a file-system type be provided
    - Others inspect the structures and determine the file-system type
  - The OS verifies that the device contains a valid file system
    - Ask the device driver to read the device directory
    - Verify that the directory has the expected format
  - The OS notes in its directory structure that a file system is mounted at the specified mount point
    - Enable the operating system to traverse its directory structure and switch among file systems

File-System Mounting (2/2)

- Systems impose semantics to clarify functionality
  - Example 1
    - Disallow a mount over a directory that contains files
    - Obscure the directory's existing files until the file system is unmounted
  - Example 2
    - Allow multiple mounts (at different mount points) per file system
    - Only allow one mount per file system



Outline

- ❑ File Systems
- ❑ File-System Mounting
- ❑ **Partitions and Mounting**
- ❑ File Sharing
- ❑ Virtual File Systems
- ❑ Remote File Systems
- ❑ Consistency Semantics
- ❑ NFS

Partitions and Mounting (1/2)

- Each partition can be either **raw** (having no file system) or **cooked** (having a file system)
  - Raw disk is used where no file system is appropriate
- If a partition contains a file system that is bootable, then the partition also needs boot information
  - The information has its own format because the system does not have the file-system code loaded at boot time
    - Usually a sequential series of blocks loaded as an image into memory
  - The image, the **bootstrap loader**, knows enough about the file-system structure to be able to find and load the kernel and start executing
    - Many systems can be **dual-booted**, allowing us to install multiple operating systems on a single system

Partitions and Mounting (2/2)

- The **root partition** selected by the boot loader is mounted at boot time
  - Contain the operating-system kernel and sometimes other system files
- Mount operation
  - The operating system verifies that the device contains a valid file system
    - Ask the device driver to read the device directory
    - Verify that the directory has the expected format
  - If the format is invalid, the partition must have its consistency checked and possibly corrected
  - The operating system notes in its in-memory mount table that a file system is mounted, along with the type of the file system

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- **File Sharing**
  - Multiple Users
- Virtual File Systems
- Remote File Systems
- Consistency Semantics
- NFS

File Sharing

- To implement sharing and protection, the system must maintain more file and directory attributes
  - Most systems have evolved to use the concepts of file (or directory) **owner** (or **user**) and **group**
    - The owner is the user who can change attributes and grant access and who has the most control over the file
    - The group attribute defines a subset of users who can share access to the file
  - The owner and group IDs of a given file (or directory) are stored with the other file attributes
    - When a user requests an operation on a file, the user ID can be compared to determine if the requesting user is the owner of the file
    - Likewise, the group IDs can be compared
    - The result indicates which permissions are applicable
    - The system then applies those permissions to the requested operation

Outline

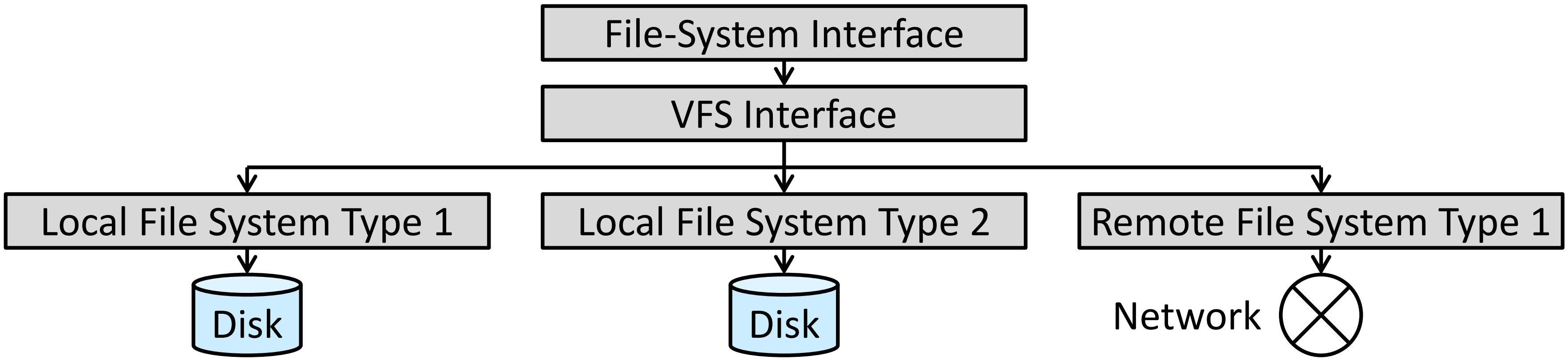
- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- **Virtual File Systems**
- Remote File Systems
- Consistency Semantics
- NFS

Multiple Types of File Systems

- An obvious but suboptimal method is to write directory and file routines for each type
- Most operating systems use object-oriented techniques to simplify, organize, and modularize the implementation
  - They allow very dissimilar file-system types to be implemented within the same structure, including network file systems, such as NFS
  - Users can access files contained within multiple file systems on the local drive or even on file systems available across the network

Virtual File Systems

- File-system interface based on the **`open()`**, **`read()`**, **`write()`**, and **`close()`** calls and on file descriptors
- **Virtual file system** (VFS) layer
  - Separate file-system-generic operations from their implementation by defining a clean VFS interface
  - Provide a mechanism for uniquely representing a file throughout a network
    - The VFS is based on a file-representation structure, called a **vnode**
    - vnode contains a numerical designator for a network-wide unique file
    - This networkwide uniqueness is required for support of network file systems



Virtual File Systems: Linux

- Four object types
  - **inode object**: represents an individual file
  - **File object**: represent an open file
  - **Superblock object**: represent an entire file system
  - **Dentry object**: represent an individual directory entry
- For each type, the VFS defines a set of operations
  - Every object of one of these types contains a pointer to a function table
  - The function table lists the addresses of the actual functions that implement the defined operations for that particular object
  - Examples for the file object
    - `int open(...)`: open a file
    - `int close(...)`: close an already-open file
    - `ssize_t read(...)` / `write(...)`: read/write from a file
    - `int mmap(...)`: memory-map a file

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- **Remote File Systems**
  - The Client-Server Model
  - Distributed Information Systems
  - Failure Models
- Consistency Semantics
- NFS

Remote File Systems

- Networking allows the sharing of data in the form of files
  - The first implemented method is **`ftp`**
    - It involves manually transferring files between machines via programs
    - **`ftp`** is used for both anonymous and authenticated access
  - The second major method uses a **distributed file system** (DFS)
    - Remote directories are visible from a local machine
    - Its involves a much tighter integration between the machine that is accessing the remote files and the machine providing the files
  - The third method is the **World Wide Web**
    - A browser is needed to gain access to the remote files
    - Separate operations (essentially a wrapper for **`ftp`**) are used to transfer files

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- **Remote File Systems**
  - **The Client-Server Model**
  - Distributed Information Systems
  - Failure Models
- Consistency Semantics
- NFS

The Client-Server Model

- Remote file systems allow a computer to mount one or more file systems from one or more remote machines
  - The machine containing the files is the server
  - The machine seeking access to the files is the client
- Security results in many challenges
  - Example: a client can be specified by a network name or other identifiers, but these can be **spoofed** or imitated
  - In the case of UNIX and its network file system (NFS), authentication takes place via the client networking information, by default
- File operation requests are sent on behalf of the user across the network to the server via the DFS protocol
  - The server determines if the user has credentials to access the file in the mode requested

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- **Remote File Systems**
  - The Client-Server Model
  - **Distributed Information Systems**
  - Failure Models
- Consistency Semantics
- NFS

Distributed Information Systems

- Provide unified access to the information needed for remote computing, also known as **distributed naming services**
  - The **domain name system** (DNS) provides host-name-to-network-address translations
  - The **network information service** (NIS) provides user-name, password, user ID, group ID space for a distributed facility
  - In Microsoft's **common Internet file system** (CIFS), network information is used with user authentication to create a network login request
    - **Active directory** as a distributed naming structure
    - **Kerberos** network authentication protocol
  - **Lightweight directory-access protocol** (LDAP) as a secure distributed naming mechanism

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- **Remote File Systems**
  - The Client-Server Model
  - Distributed Information Systems
  - **Failure Models**
- Consistency Semantics
- NFS

Failure Models

- Local file systems can fail for a variety of reasons
  - Failure of the drive containing the file system
  - Corruption of the directory structure or other disk-management information (metadata)
  - Disk-controller, cable, or host adapter failure
  - User or system-administrator failure
- Remote file systems have even more failure modes
  - Network or server failure, networking implementation issues
- If both server and client maintain **state information**, then they can seamlessly recover from a failure
  - NFS v3 implements a **stateless** DFS, carrying all the information needed, which is resilient, rather easy to implement, and unsecure
  - NFS v4 is made stateful to improve its security, performance, and functionality

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- **Consistency Semantics**
  - UNIX Semantics
  - Session Semantics
  - Immutable-Shared-Files Semantics
- NFS

Consistency Semantics

- An important criterion for evaluating any file system that supports file sharing
  - Specify how multiple users of a system are to access a shared file simultaneously
  - Specify when modifications of data by one user will be observable by other users
- These semantics are typically implemented as code with the file system
- The series of accesses between the **`open()`** and **`close()`** operations makes up a **file session**

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- **Consistency Semantics**
  - **UNIX Semantics**
  - Session Semantics
  - Immutable-Shared-Files Semantics
- NFS

UNIX Semantics

- The UNIX file system uses the following consistency semantics
  - Writes to an open file by a user are visible immediately to other users who have this file open
  - One mode of sharing allows users to share the pointer of current location into the file
    - The advancing of the pointer by one user affects all sharing users
    - A file has a single image that interleaves all accesses, regardless of their origin
- A file is associated with a single physical image that is accessed as an exclusive resource
  - Contention for this single image causes delays in user processes

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- **Consistency Semantics**
  - UNIX Semantics
  - **Session Semantics**
  - Immutable-Shared-Files Semantics
- NFS

Session Semantics

- The Andrew file system (OpenAFS) uses the following consistency semantics
  - Writes to an open file by a user are not visible immediately to other users that have the same file open
  - Once a file is closed, the changes made to it are visible only in sessions starting later
    - Already open instances of the file do not reflect these changes
- A file may be associated temporarily with several images at the same time
  - Multiple users are allowed to perform both read and write accesses concurrently on their images of the file, without delay

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- **Consistency Semantics**
  - UNIX Semantics
  - Session Semantics
  - **Immutable-Shared-Files Semantics**
- NFS

Immutable-Shared-Files Semantics

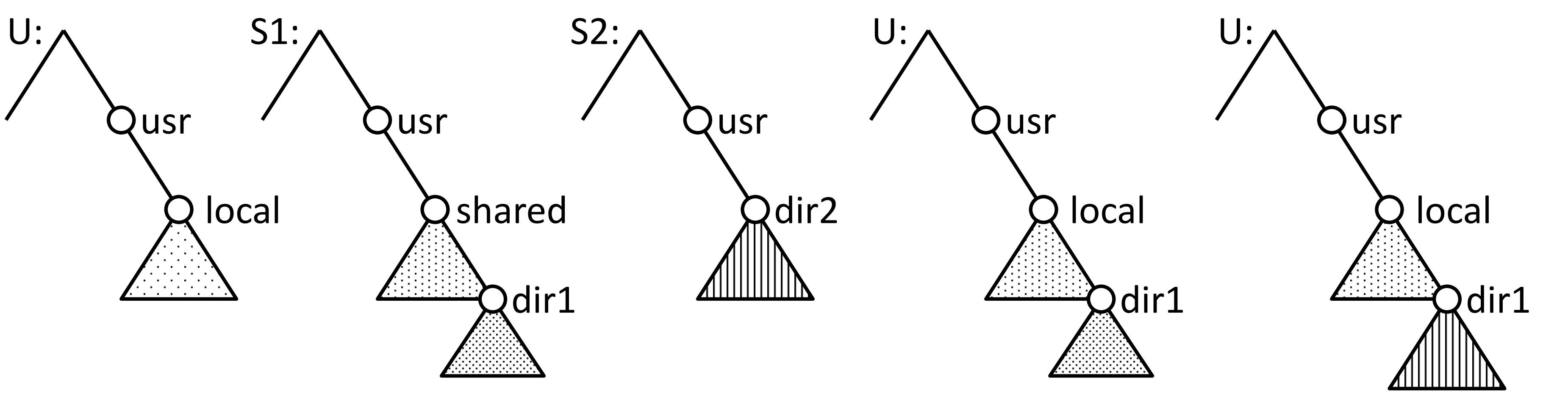
- Once a file is declared as shared by its creator, it cannot be modified
  - An immutable file has two key properties
    - Its name may not be reused
    - Its contents may not be altered
  - The name of an immutable file signifies that the contents of the file are fixed
- The implementation of these semantics in a distributed system is simple
  - The sharing is disciplined (read-only)

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- Consistency Semantics
- **NFS (Network File System)**
  - The Mount Protocol
  - The NFS Protocol
  - Path-Name Translation
  - Remote Opertions

NFS (1/2)

- An implementation and a specification of a software system for accessing remote files across LANs (or WANs)
  - NFS views a set of interconnected workstations as a set of independent machines with independent file systems
  - Sharing is based on a client-server relationship
- Mounting
  - Diskless workstations can even mount their own roots from servers
  - Cascading mounts are also permitted in some NFS implementations



Three Independent File Systems

Mounts

Cascading Mounts

NFS (2/2)

- NFS is to operate in a heterogeneous environment of different machines, operating systems, and network architectures
  - The NFS specification is independent of these media
    - This independence is achieved through the remote procedure call (RPC) primitives built on top of an external data representation (XDR) protocol
- The NFS specification distinguishes between
  - The services provided by a mount mechanism
    - A mount protocol
  - The actual remote-file-access services
    - A protocol for remote file accesses, the NFS protocol

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- Consistency Semantics
- **NFS**
  - **The Mount Protocol**
  - The NFS Protocol
  - Path-Name Translation
  - Remote Operations

The Mount Protocol

- Establish the initial logical connection between a server and a client
  - A mount operation includes
    - The name of the remote directory to be mounted
    - The name of the server machine storing it
  - The mount operation changes only the user's view and does not affect the server side
  - The mount request is mapped to the corresponding RPC and is forwarded to the mount server running on the specific server machine
    - The server maintains an **export list** that specifies local file systems that it exports for mounting, along with names of permitted machines
  - If request conforms to the export list, the server returns to the client a file handle for accesses to files within the mounted file system
    - The file handle contains all the information that the server needs to distinguish an individual file it stores

Outline

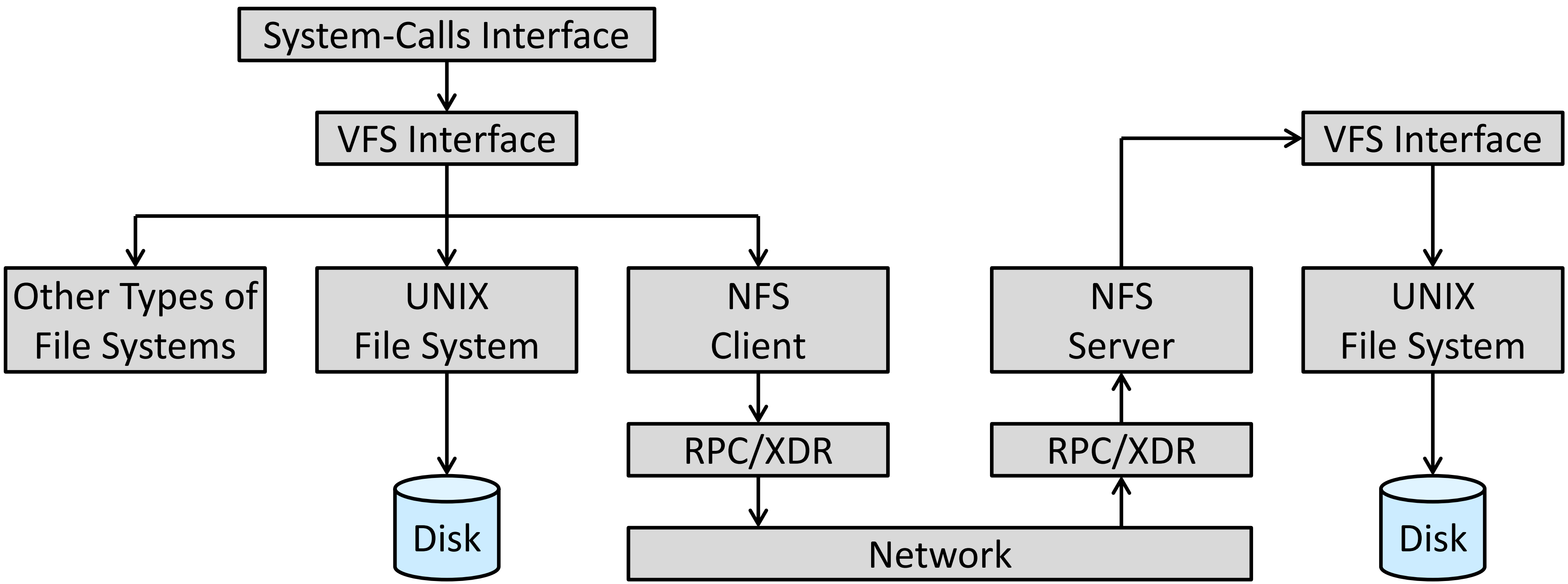
- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- Consistency Semantics
- **NFS**
  - The Mount Protocol
  - **The NFS Protocol**
  - Path-Name Translation
  - Remote Operations

The NFS Protocol

- Provide a set of RPCs for remote file operations
  - Search for a file within a directory
  - Read a set of directory entries
  - Manipulate links and directories
  - Access file attributes
  - Read and write files
- These procedures can be invoked only after a file handle for the remotely mounted directory has been established
- NFS v3 is stateless (NFS v4 is stateful)
  - Servers do not maintain information about clients from one access to another
  - Each request has to provide a full set of arguments
  - Modified data must be committed to the server's disk before returned

Schematic View of NFS Architecture

- NFS is integrated into the operating system via a VFS
  - How an operation on an already-open remote file is handled



Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- Consistency Semantics
- **NFS**
  - The Mount Protocol
  - The NFS Protocol
  - **Path-Name Translation**
  - Remote Operations

Path-Name Translation

- Break the path name into the names of separate directory entries or components
  - Perform a separate NFS `lookup` call for every pair of component name and directory vnode
- Speed up references to files with the same initial path name
  - A directory-name-lookup cache on the client side holds the vnodes for remote directory names

Outline

- File Systems
- File-System Mounting
- Partitions and Mounting
- File Sharing
- Virtual File Systems
- Remote File Systems
- Consistency Semantics
- **NFS**
  - The Mount Protocol
  - The NFS Protocol
  - Path-Name Translation
  - **Remote Operations**

Remote Operations

- Almost one-to-one correspondence between the regular UNIX system calls for file operations and the NFS protocol RPCs
  - Except opening and closing files
- In practice, buffering and caching are employed for performance
  - When a file is opened, the kernel checks with the remote server to determine whether to fetch or revalidate the cached attributes
    - The file-attribute (inode-information) cache is updated whenever new attributes arrive from the server
    - The file-blocks cache is used only if the cached attributes are up to date
- Both read-ahead and delayed-write techniques are used
  - Clients do not free delayed-write blocks until the server confirms that the data have been written to disk

Objectives

- Delve into the details of file systems and their implementation
- Explore booting and file sharing
- Describe remote file systems, using NFS as an example

Q&A