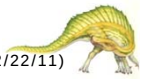
Chapter 11: File System Implementation

- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management
- Efficiency and Performance
- Recovery
- Log-Structured File Systems
- NFS
- Example: WAFL File System

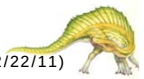
(slides selected/reordered/improved by R. Doemer, 02/22/11)



Efficiency and Performance

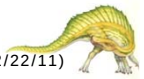
- **Efficiency** depends on:
  - Disk allocation and directory algorithms
  - Type of data kept in file's directory entry
- **Performance** improvement techniques:
  - **Disk cache**
    - Separate section of main memory (in kernel space) for frequently used disk blocks
  - **Virtual disk** (RAM disk)
    - Dedicate a section of main memory as virtual file-system
  - **Free-behind** and **read-ahead** techniques
    - Optimization for sequential access

(slide improved by R. Doemer, 02/22/11)



Backup and Recovery

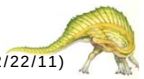
- **System failure** (e.g. sudden power outage) may result in
  - Loss of data
  - Inconsistency of data
- **File system recovery** techniques
  - **Consistency checker**
    - Compares data in directory structure with data blocks on disk, and tries to fix inconsistencies
    - Examples: **`fsck`** in Unix, **`chkdsk`** in Windows
  - **Back up**
    - Use system programs to *regularly* back up data from disk to another storage device (e.g. magnetic tape or other disk)
    - Recover lost file or disk by **restoring** data from backup



(slide improved by R. Doemer, 02/22/11)

Log Structured File Systems

- **Log-based transaction-oriented** file systems
  - Record each update to the file system as a **transaction**
  - aka. **journaling** file system
- All transactions are written to a **log** file
  - A transaction is considered **committed** once it is written to the log
  - However, the file system may not yet be updated
- Transactions in the log are *asynchronously* written to the file system
  - When the file system is successfully modified, the transaction is removed from the log
- If the file system crashes, all remaining transactions in the log must still be performed

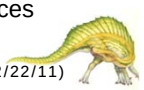


(slide improved by R. Doemer, 02/22/11)

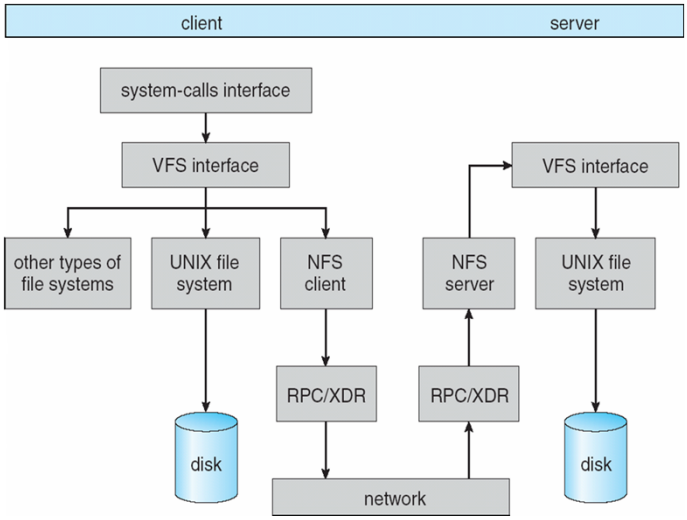
Network File System (NFS)

- The Sun **Network File System (NFS)**
  - An implementation and a specification of a software file system for accessing remote files across **LAN**s (or **WAN**s)
  - The implementation is part of the *Solaris* and *SunOS* operating systems running on Sun workstations
    - Now available also for most other OS
- **NFS**
  - is built on top of the *unreliable datagram protocol* (UDP/IP protocol)
    - e.g. over Ethernet
  - is designed to operate in a *heterogeneous environment* of different machines, operating systems, and network architectures
    - NFS specification is independent of these media
  - uses *remote procedure call* (RPC) primitives between two implementation-independent file system interfaces

(slide improved by R. Doemer, 02/22/11)

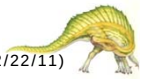


Schematic View of NFS Architecture



NFS Architecture Layers

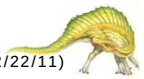
- **UNIX file-system** interface layer
  - Based on *open*, *read*, *write*, and *close* calls, and *file descriptors*
- **Virtual File System** (VFS) layer
  - Activates file-system-specific operations to handle requests according to the file-system types
  - Calls the NFS protocol procedures for remote requests
- **NFS service** layer
  - Bottom layer of the architecture
  - Implements the NFS protocol
    - Based on remote procedure calls (RPC)



(slide improved by R. Doemer, 02/22/11)

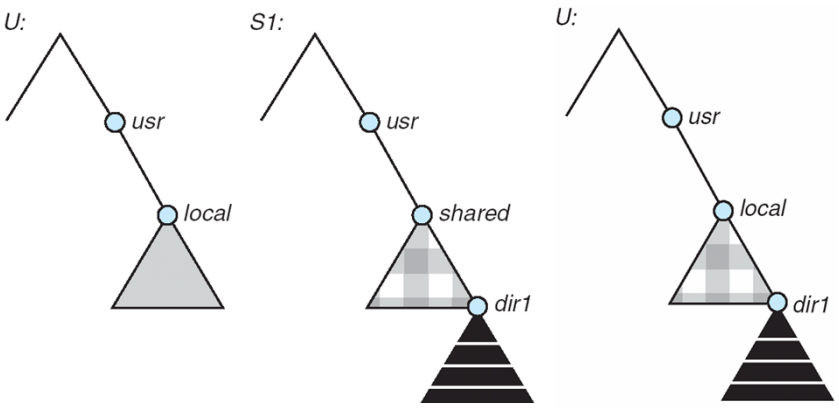
NFS Mounting

- **Interconnected workstations** are viewed as a set of *independent machines* with *independent file systems*
  - A **remote directory** is *mounted* over a local file system directory
    - The mounted directory looks like an integral sub-tree of the local file system (*transparent* to the user)
    - Unless empty, it *replaces* the sub-tree descending from the local directory
  - Specification of the remote directory for the **mount operation** is *non-transparent* (for the system administrator)
    - Host and full name of the remote directory have to be provided
    - Files in the remote directory can then be accessed in a transparent manner



(slide improved by R. Doemer, 02/22/11)

NSF Mounting



Local file system (on NFS client)

Remote file system (on NSF server)

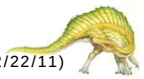
Remote sub-tree mounted on client

(slide improved by R. Doemer, 02/22/11)

NFS Mount Protocol

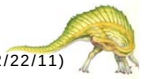
- The NFS specification distinguishes between
  - the services provided by a **mount mechanism** and
  - the actual **remote-file-access** services
- **NFS mount protocol**
  - establishes an **initial logical connection** between *server* and *client*
  - **Mount operation** includes name of remote 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*, 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)
- The *mount operation changes only the user's view* and does not affect the server side

(slide improved by R. Doemer, 02/22/11)



NFS Access Protocol

- Provides a set of **remote procedure calls** for remote file operations
- The procedures support the following 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
  - Modified data must be committed to the server's disk before results are returned to the client
- The NFS protocol does not provide concurrency-control mechanisms



(slide improved by R. Doemer, 02/22/11)

End of Chapter 11