Western

UNIVERSITY · CANADA

Chapter 15 – File-System Internals

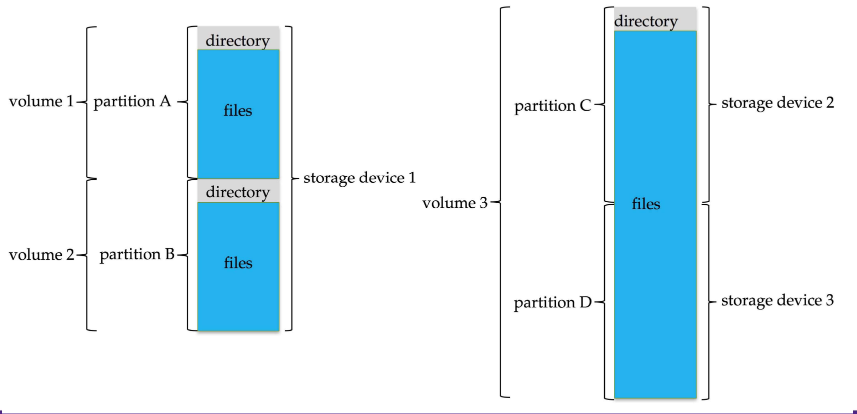
Spring 2023

Overview

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

File Systems

- General-purpose computers can have multiple storage devices
- Multiple storage devices can be sliced into partitions
- Partitions are assigned to one or more volumes
- Each volume usually formatted into a file system



File Systems

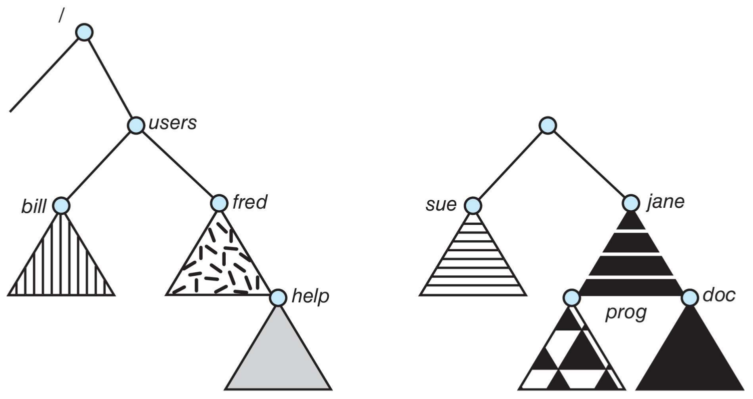
- Common general-purpose file systems
  - UNIX/Linux – ext2, ext3, ext4, zfs, ufs
  - Windows – NTFS, FAT, FAT32
- Special-purpose file systems
  - tmpfs for temporary I/O
  - procfs for presenting process information as a file system in UNIX/Linux
  - Other file systems for specific workloads (e.g. Google File System)

File-System Mounting

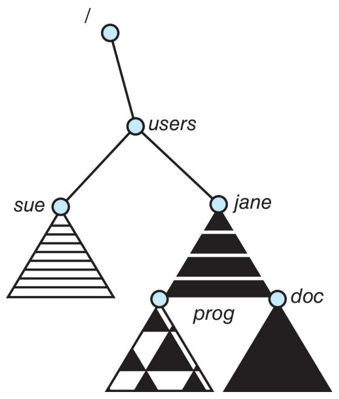
- Just as files must be opened before they can be read, a file system must be **mounted** before it can be used
- The operating system is given the name of the device and a **mount point**. The mount point is a location within a file structure (typically an empty directory)
    - The file system may be provided, or it may be automatically determined
- If the file system is already mounted, the operating system must decide whether to allow multiple mounts or not
- If the mount point is not empty, the operating system must decide whether to prevent the action or to obscure the existing files and directories until the file system is unmounted again

File-System Mounting

- E.g. Mounting a file system to a directory that is not empty



(a) (b)



Partitions and Mounting

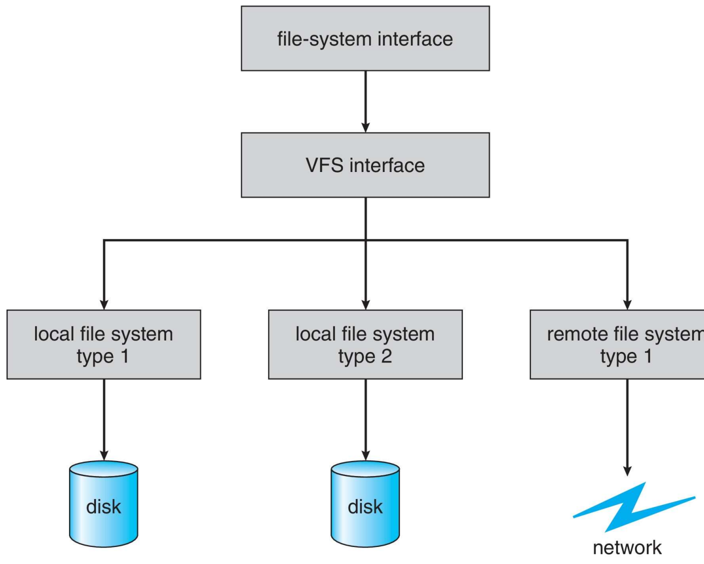
- A partition can be
    - **Raw** – No file system. Just blocks. Suitable for swap space or custom workloads (e.g. Databases, RAID)
    - **Cooked** – Contains a structured file system
        - Bootable partition – Contains enough code to find and mount other file systems. If it can understand multiple operating systems, then the system can be **dual-booted**
        - Root partition – Contains the operating system and the ability to mount other partitions

File Sharing

- User oriented operating systems must accommodate the need to share files
- File permission metadata is kept in the FCB
- The operating system uses the same permissions across all its file systems
- Care must be taken when mounting a file system to another operating system (over the network or by physically plugging disks into other systems)
    - E.g. What if `jsmith` has user id `1001` on one system but user id `1002` on another?

Virtual File Systems

- To make using multiple file systems seamless, it is useful to separate file-system generic operations from implementation details
- The virtual file system will then dispatch the correct operation to the file-system



Remote File Systems

- Early computing used protocols such as `ftp` to transfer files between machines
- Later implementations employ a **distributed file system (DFS)** allowing remote file systems to be visible on local machines
- File transfers across the Internet are now common using the `http` protocol

Remote File Systems

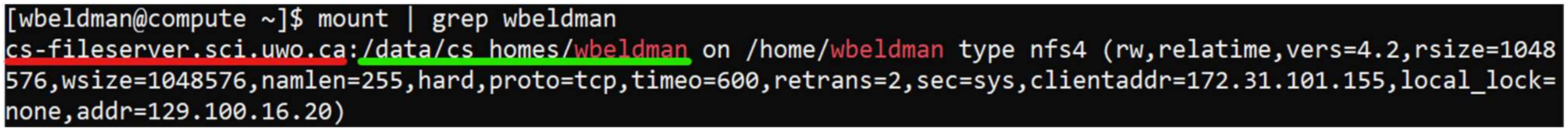
- DFS client-server model
  - **Client** – The machine seeking to mount the remote file system
  - **Server** – The machine sharing the local file system
- The server must employ a mechanism to authenticate and authorize clients
- The client typically sends user id information along with standard operations
- Remote file systems need to account for more failure conditions than local file systems
  - Keeping state on both the client and server side can help identify when errors occur. This is also more secure, but it adds overhead.

Consistency Semantics

- Remote file systems must specify behaviour when multiple users share a file simultaneously
- Process synchronization techniques seen in Chapter 6 would generate too much traffic on the network. A simpler approach is needed.
- **File session** – All the actions between the `open()` and the `close()` call
- The server may block client requests until the file session completes
- Alternatively, the file system could be shared read-only

NFS

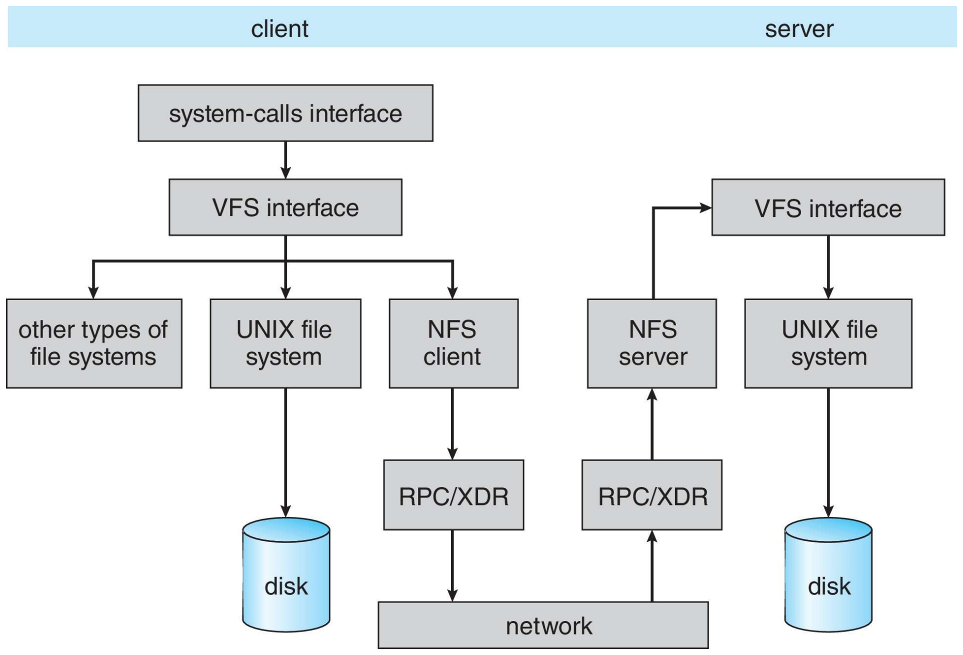
- **Network File System (NFS)** is a popular remote file system that employs a client-server model.
- Windows more commonly uses **common Internet file system (CIFS)**
- The NFS client mounts a remote NFS share. The share appears as though it is part of the local file system
- NFS is designed to work in heterogeneous environments (different operating systems, different machines, different networking protocols, etc.)



```
[wbeldman@compute ~]$ mount | grep wbeldman
cs-fileserver.sci.uwo.ca:/data/cs_homes/wbeldman on /home/wbeldman type nfs4 (rw,relatime,vers=4.2,rsize=1048
576,wsize=1048576,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.31.101.155,local_lock=
none,addr=129.100.16.20)
```

NFS

- The NFS architecture



NFS

- The NFS mount protocol
  - The client requests a remote directory
  - The server verifies the client has permission to mount the directory including read-only or read-write permissions
  - If permission is granted, the mount is recorded in an export table.
    - The export table can be used to track all mounts, notify clients of downtime, revoke access if necessary, etc.

NFS

- The NFS remote operations protocol
  - A set of **remote procedure calls (RPCs)** to handle
    - 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

- The NFS remote operations protocol
  - NFS version 3 is stateless so all operations must provide all arguments
    - This requires no additional overhead on the client or server side so it is faster but less secure
  - NFS version 4 is stateful. It has additional overhead but is more secure
  - All versions must safely write data to disk so there may be delays up to seconds or minutes depending on other clients and other file sessions

Western

UNIVERSITY · CANADA