File systems

CS503: Operating systems, Spring 2019

Pedro Fonseca
Department of Computer Science
Purdue University

Previous lecture

- Xinu offer low-level message passing
    - Only one message outstanding per process
- Xinu high-level message passing
    - Dynamically created ports
    - Number of messages and size fixed when port created
    - Arbitrary senders an receivers
    - Synchronous interface
- Port reset/deletion is tricky because of concurrency
- Linux IPC
- Shared memory vs. message passing; build on top of Xinu ports; etc.

What is a file system?

- A way to organize variable-size persistent data
    - Organize, store, retrieve, delete information
- Defines the structure and implements the operations

FS challenges and requirements

- Huge files
- Many files
- Support the backing devices: magnetic tapes, HDD, SDD, non-volatile memory, non-volatile
- Data may need to survive the process lifetime
- Consistency
- Data revisiting, check-pointing
- Distributed remote access

Durability

- Stored information must survive the termination of the process using it
- Lifetime can be seconds to years

Storage terminology

- Disk: Non-volatile block-addressable storage
- Block: Smallest unit of IO on a disk
    - Common block size = 512 bytes or 4KB
- Partition: A set of contiguous blocks on a disk

File system terminology

- File: A unit of data managed by the file system
- Data: The user data associated with a file
    - Unstructured (byte stream) or structured (records)
- File name:
    - A textual name that identifies the file

File system terminology

- Metadata: Information about the file (e.g., owner, creation time, permission, length of file data, location of file data, etc.)
    - As opposed to the data in the file
- Directory (folder):
    - Container for the file names
    - Directories within directories provide a hierarchical name system

File system terminology

- Superblock:
  - A record of file system's properties, containing the key file system information
- Inode (file control block):
  - A structure that stores a file's metadata and location of file data

Design choices

- Namespaces: flat, hierarchical, or other
- File system types: support one type of filesystem or multiple types
- File types: unstructured (byte streams) or structured (indexed files)
- Metadata: What kind of attribute are should be stored?
- Implementation: How is data laid out on disk?

Working with OS File system operations

Formatting

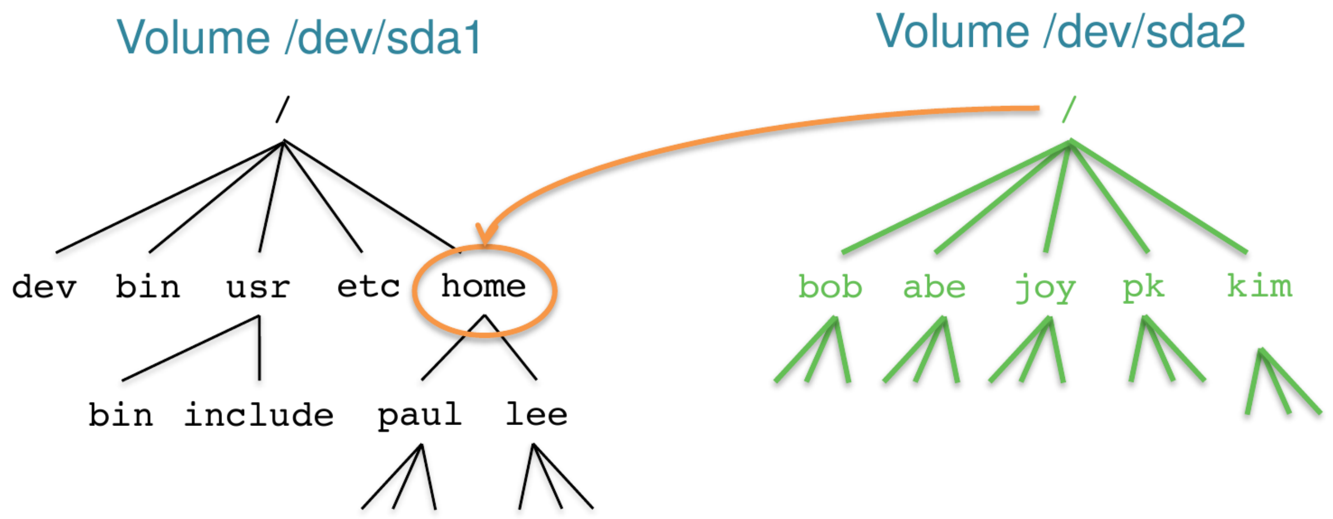
- Partitioning: Divide a disk into one or more regions. Each can hold a separate file system
- High-level formatting: initialize a file system for use
- Initializing a file system:
  - Determine where various data structures will be stored
    - Free block bitmaps, inode lists, data blocks
    - Initialize structures to reflect an empty file system

Mounting

- Make file system available for use
- **Mount** system call (in Linux):
  - Args: file system type, block device, mount point
- Steps:
  - Access the raw disk (block device)
  - Read superblock and file system metadata
  - Prepare in-memory data structures:
    - In-memory version of the superblock
    - References to the root directory

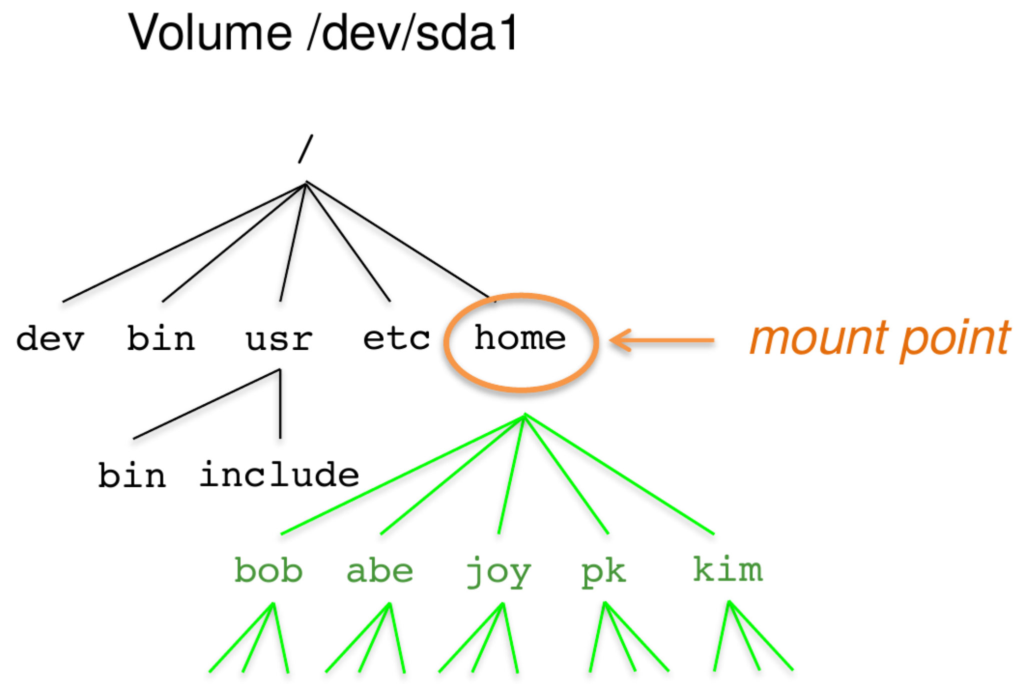
Mounting: building up a name space

- Combine multiple file systems into a single hierarchical name space
- The mounted file overlays (i.e., hides) anything in the file system under that mount point
- Looking up a pathname may involve traversing multiple mount points



Mounting: building up a name space

- Result



`/home/paul` and `/home/lee` are no longer visible

Creating a file

- Create an inode to hold info (metadata) about the file
    - Initialize timestamps
    - Set owner/permissions/modes
- Add a directory entry (dentry) for the new inode
    - Directory entry holds: a set of {filename, inode #}
    - Use current directory or pathname specified by filename

Creating a directory

- A directory is just like a file:
    - Contents = set of {name, inode} pairs
- Steps:
    - Create a new inode and initialize it
    - Initialize contents to contain:
        - A directory entry to the parent (name = “..”)
        - A directory entry to itself (name = “.”)

Open a file

- Lookup: scan one or more directories to find the name
    - Pathname traversal
    - Mount point traversal
- Get info and verify ownership & access permissions
    - Read the inode (found through the directory entry)
    - Check ownership and access permissions
- Return a file handle (file descriptor)
    - Index into an open files table for the process

Write to a file

- OS keeps track of current read/write offset of each open file (seek pointer)
  - Can be modified with lseek Linux system call
- Basic steps:
  - Read file data if not writing on a block boundary
  - Write one or more blocks of data from memory to disk
  - Update file size
  - Update the current file offset in memory
- Writes are usually buffered in memory and delayed to optimize performance

Read from a file

- Basic steps:
    - Check size of file to ensure no read past end of file
    - Identify one or more blocks that need to be read
        - Information is in inode usually cached in memory
    - May need to read additional blocks to get the block map to find the desired block number
    - Increment the current file offset by the amount that was read

Delete a file

- Remove the file from its directory entry
    - This stops other programs from opening it: they won’t see it
- If there are no program with open references to the file
    - Mark all the blocks used by the file as free
    - Mark the inode used by the file as free
    - Check this condition when closing a file (or existing a process)

Read a directory

- Directories are like files but contain a set of {name, inode} tuples
- The FS implementation parses this storage structure
- Operations (in Linux):
  - opendir
  - readdir
  - closedir

Read and write metadata

- Read inode information: stat Linux system call
- Write metadata: calls to change specific properties
  - chown
  - chgrp
  - chmod
  - utime
- Extended attributes (name-value sets)
  - listxattr
  - getxattr
  - setxattr
  - removexattr

Summary

- Storage terminology
- File system terminology
- Design space
- inodes and metadata
- Typically FS interface and basic steps
    - Mounting file system
    - Open
    - Read/Write
    - Delete
    - Finding files