Today's Objectives

- AWS/MR Review
- Final Projects
- Distributed File Systems

Inverted Index

- "final" input files have been posted
- Another email out to AWS
- Google cloud

FINAL PROJECT

Final Project: Schedule

- Project proposal: November ??
- Dec 4 – 8: Team Presentations
- Dec 11—15 (finals week): Write up due

Final Project: Logistics

- Work in teams of 2-4
- Choose your own project
  - See course final project page (to be posted) for some ideas
  - Using Amazon Cloud
  - Data analysis using Hadoop (or other cloud-based services)

Review

- What is RAID?
  - What is its motivation?
  - What are some variations?
  - What are the tradeoffs? When would you use one vs the other?

FILE SYSTEMS

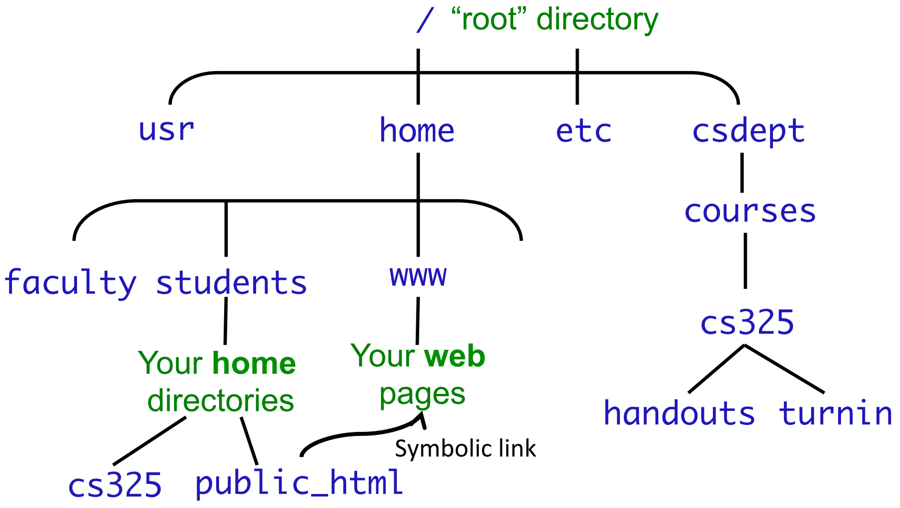
File System Characteristics

- Higher level of abstraction
  - Prevents users and processes from dealing with disk blocks and memory blocks
- File systems are responsible for:
  - Organization
  - Storage
  - Retrieval
  - Naming
  - Sharing
  - Protection

File System Characteristics

- Designed to store and manage large numbers of files
  - Create, name, delete
- Naming is supported through use of directories
  - UNIX uses pathnames to represent ***hierarchical*** naming scheme for files
- ***metadata*** refers to extra information stored by file system for managing files
  - File attributes, directories, etc

Hierarchical Naming



/ “root” directory

usr

home

etc

csdept

courses

faculty

students

www

cs325

Your **home** directories

Your **web** pages

handouts

turnin

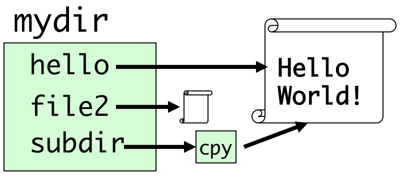
cs325

public_html

Symbolic link

Links

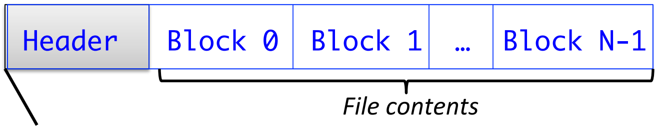
- Directories are lists of files and directories
- Each directory entry links to a file on the disk



- Two directory entries can link to the same file
  - In same directory or across different directories
- Moving a file does not actually move any data around
  - Creates link in new location
  - Deletes link in old location
- `ln` command: `ln <target> <dest>`

Files

- Contain data and attributes
  - Attributes: file length, timestamps, owner, access-control lists
- Typical File

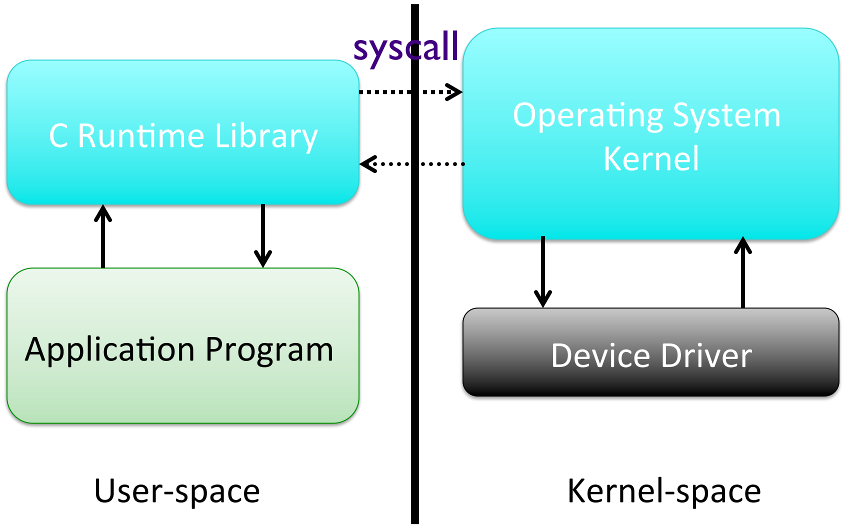


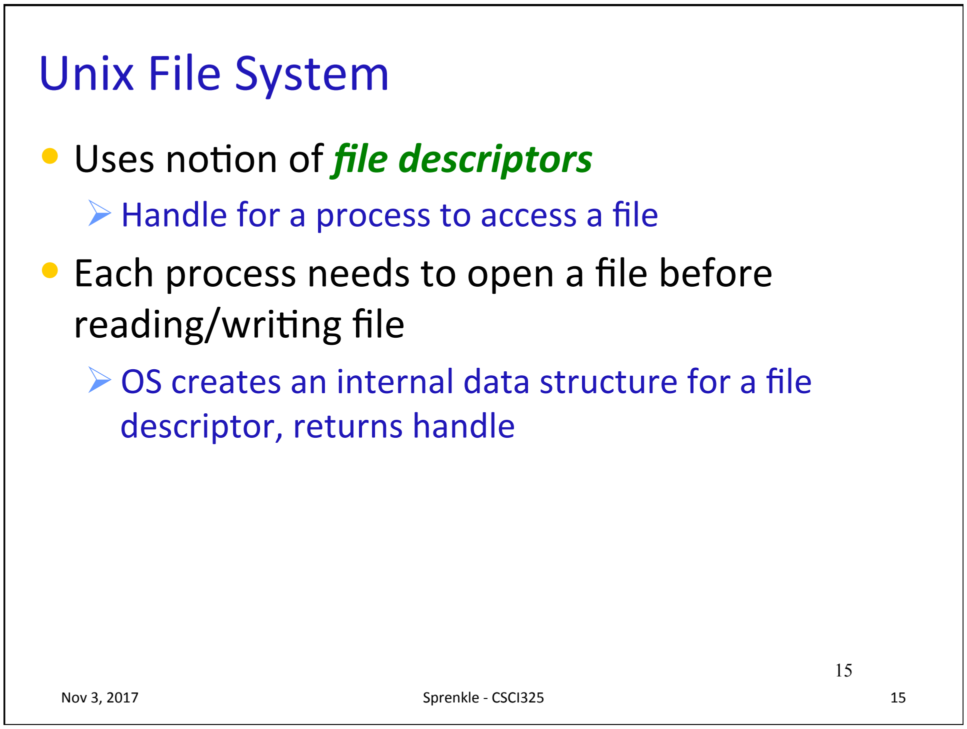
- Timestamps: creation, read, write, header
- Ownership
- Access Control List: who can access this file and in what mode
- Reference Count: Number of directories containing this file
  - May be > 1 (hard linking of files)
  - When 0, can delete file

File System Operations

- File system operations are often performed via ***system calls***
- User programs are not allowed to access system resources directly
  - must ask OS to do that on their behalf
- System calls: set of functions for user programs to request for OS services
  - Run in kernel mode
  - Invoked by special instruction causing the kernel to switch from user mode
  - When the system call finishes, processor returns to the user program and runs in user mode.

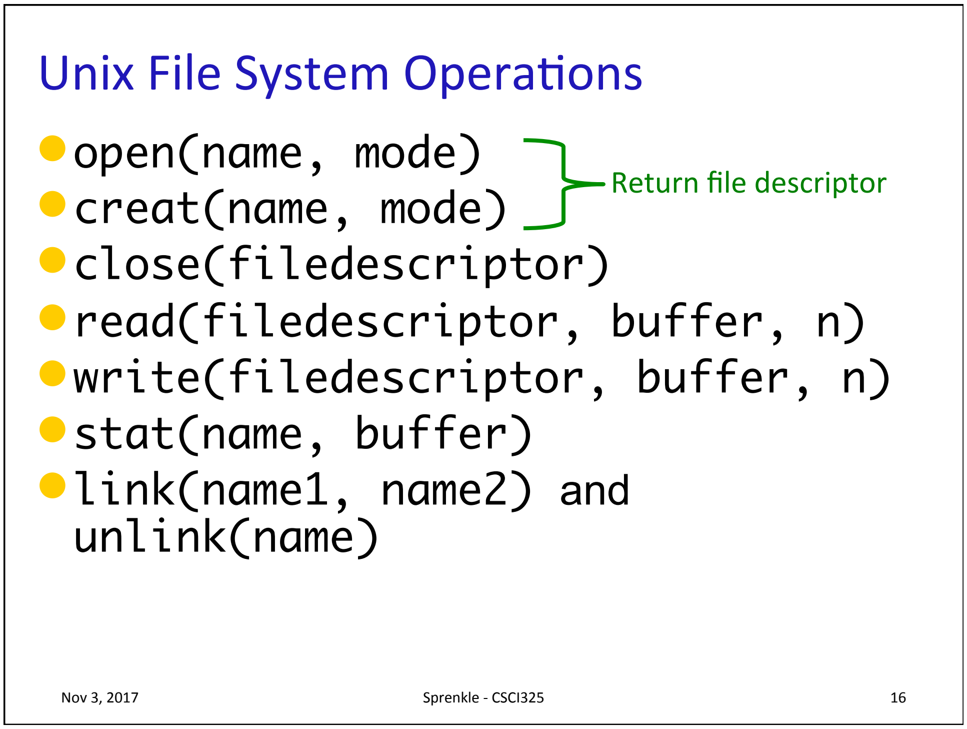
How system calls work





Unix File System

- Uses notion of ***file descriptors***
  - Handle for a process to access a file
- Each process needs to open a file before reading/writing file
  - OS creates an internal data structure for a file descriptor, returns handle



Unix File System Operations

- `open(name, mode)`
- `creat(name, mode)`

Return file descriptor

- `close(filedescriptor)`
- `read(filedescriptor, buffer, n)`
- `write(filedescriptor, buffer, n)`
- `stat(name, buffer)`
- `link(name1, name2)` and `unlink(name)`

DISTRIBUTED FILE SYSTEMS

Distributed File System

- Files are stored on a server
    - ➢ Clients communicate with server to perform operations on file
- How does the network complicate things?
- What can we do about it?

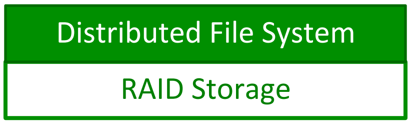
What challenges are introduced by a distributed file system (in addition to scalable storage)?

Distributed File System Requirements

- Transparency
  - Access, location, mobility, performance, scaling
- Concurrent file updates
- File replication
- Hardware and OS heterogeniety
- Fault tolerance
- Consistency
- Security
- Efficiency

Distributed File Systems and RAID

- Distributed file systems require higher reliability and capacity at higher loads than a local file system
- RAID was created to address these limitations
- Most work in distributed file systems ignores the disk-level details (like which RAID is used)
- All work assumes that the disks themselves are reliable, scalable, and have high performance which can be accomplished using RAID



Distributed File Systems: Goal & Challenges

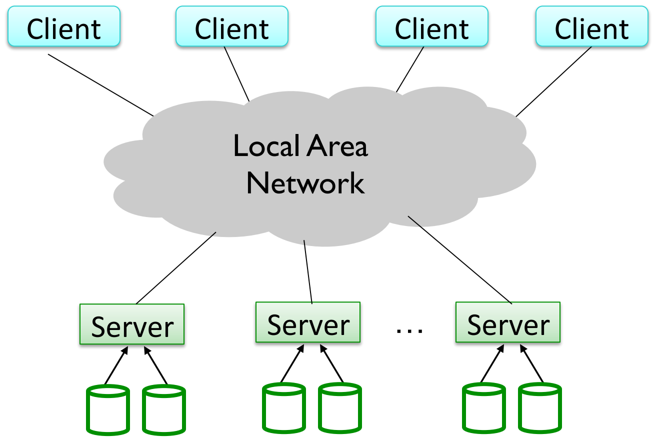
Goal: Transparent access to remote files

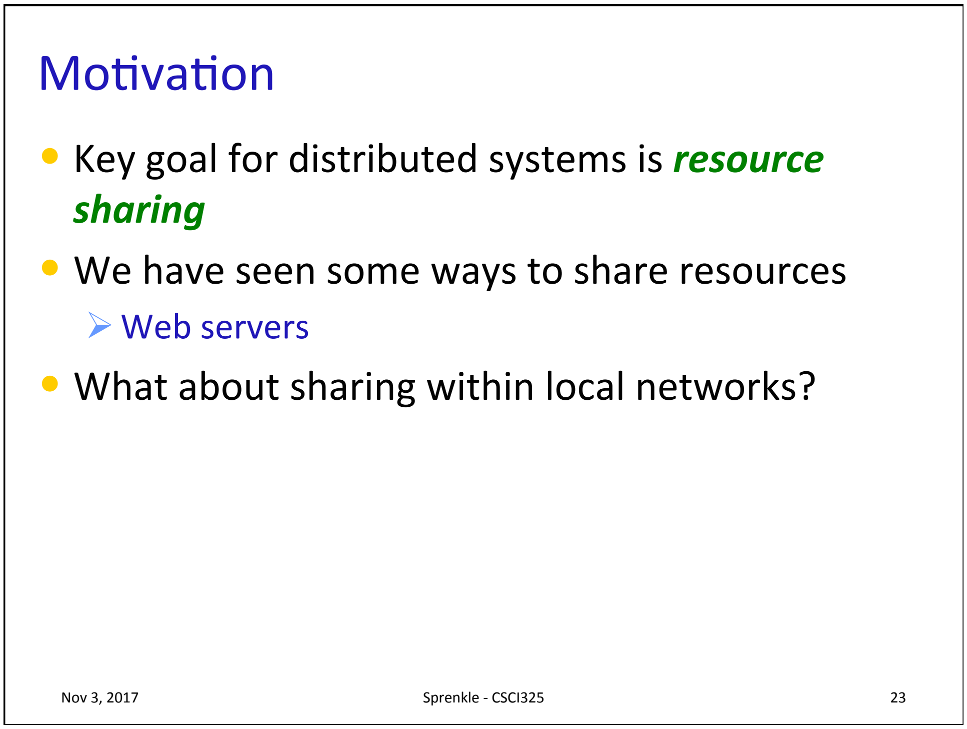
- Enable programs to store and access remote files as though they were local
- Access at any time from any computer
- Comparable performance and reliability to local disks

Why would you want this?
What are some of the hard issues?
Know any examples of DFS?

DFS: Architecture …

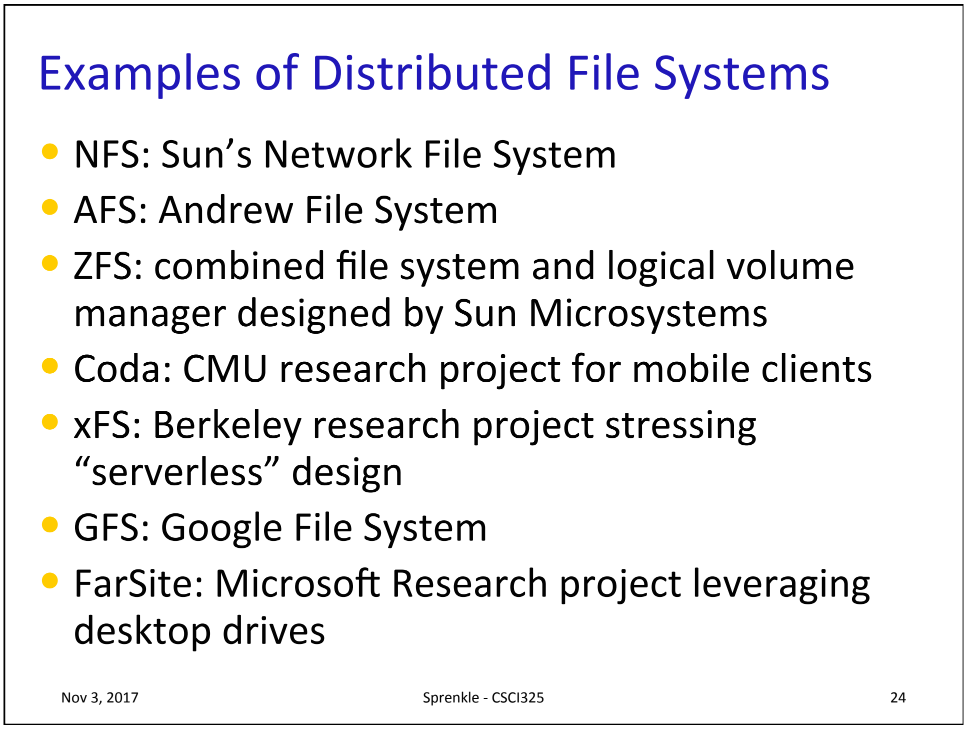
Goal: Transparent access to remote files





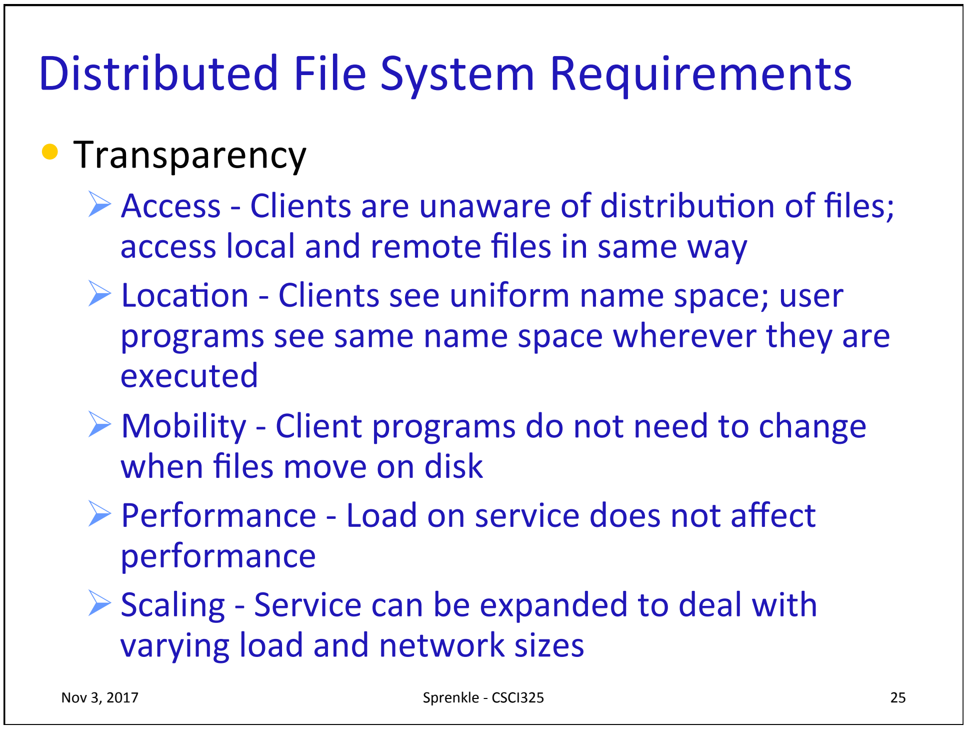
Motivation

- Key goal for distributed systems is ***resource sharing***
- We have seen some ways to share resources
  - Web servers
- What about sharing within local networks?



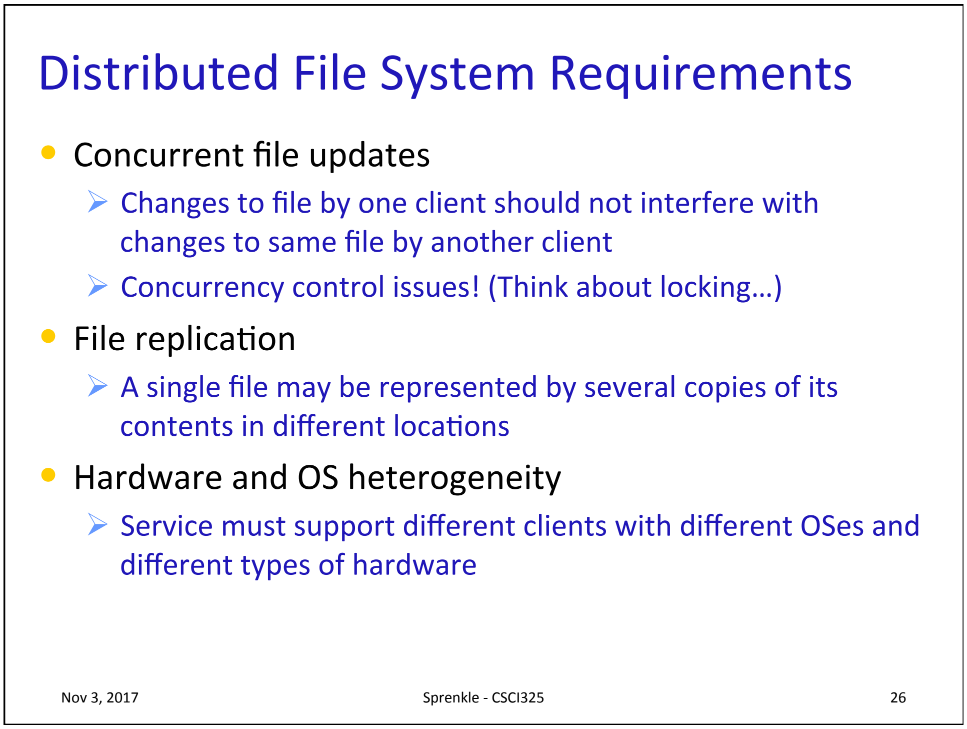
Examples of Distributed File Systems

- NFS: Sun’s Network File System
- AFS: Andrew File System
- ZFS: combined file system and logical volume manager designed by Sun Microsystems
- Coda: CMU research project for mobile clients
- xFS: Berkeley research project stressing “serverless” design
- GFS: Google File System
- FarSite: Microsoft Research project leveraging desktop drives



Distributed File System Requirements

- Transparency
  - Access - Clients are unaware of distribution of files; access local and remote files in same way
  - Location - Clients see uniform name space; user programs see same name space wherever they are executed
  - Mobility - Client programs do not need to change when files move on disk
  - Performance - Load on service does not affect performance
  - Scaling - Service can be expanded to deal with varying load and network sizes



Distributed File System Requirements

- Concurrent file updates
  - Changes to file by one client should not interfere with changes to same file by another client
  - Concurrency control issues! (Think about locking…)
- File replication
  - A single file may be represented by several copies of its contents in different locations
- Hardware and OS heterogeneity
  - Service must support different clients with different OSes and different types of hardware

Distributed File System Requirements

- Fault tolerance
  - Must operate in face of client and server failures
  - Need to support idempotent operations
- Consistency
  - Concurrent access to files that are replicated should be consistent
- Security
  - Enforce permissions and access-control lists
- Efficiency
  - Need to achieve comparable performance, power, and generality as a non-distributed file system

Looking Ahead

- Monday: Priya Mahadevan, Network Architect at Google
  - Start thinking about questions!