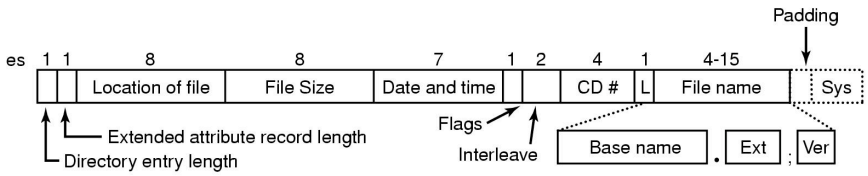
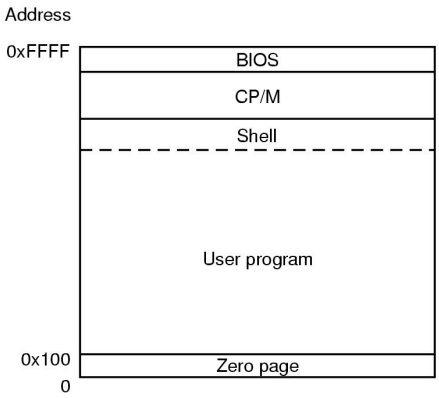
Example File Systems
CD-ROM File Systems



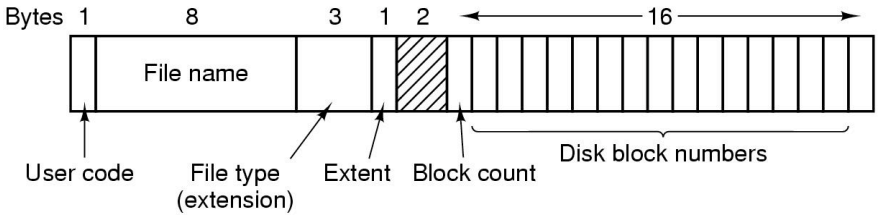
The ISO 9660 directory entry

The CP/M File System (1)



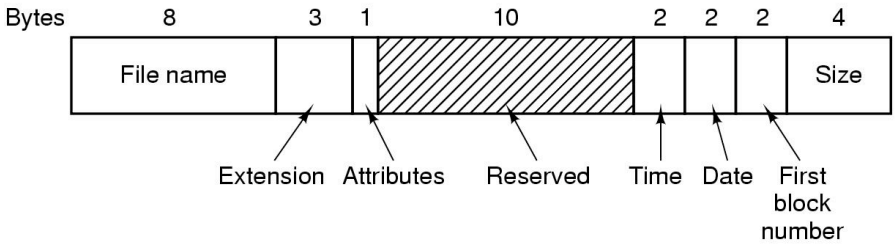
Memory layout of CP/M

The CP/M File System (2)



The CP/M directory entry format

The MS-DOS File System (1)



The MS-DOS directory entry

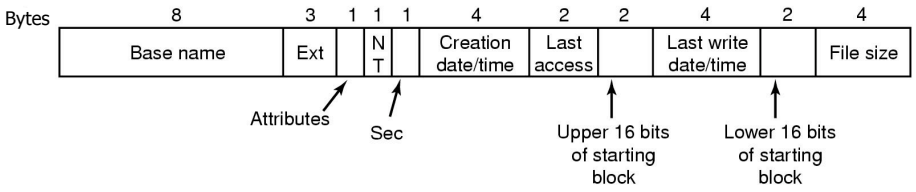
The MS-DOS File System (2)

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

- Maximum partition for different block sizes
- The empty boxes represent forbidden combinations

434

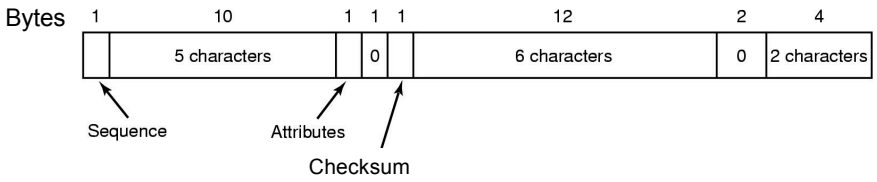
The Windows 98 File System (1)



The extended MOS-DOS directory entry used in Windows 98

435

The Windows 98 File System (2)



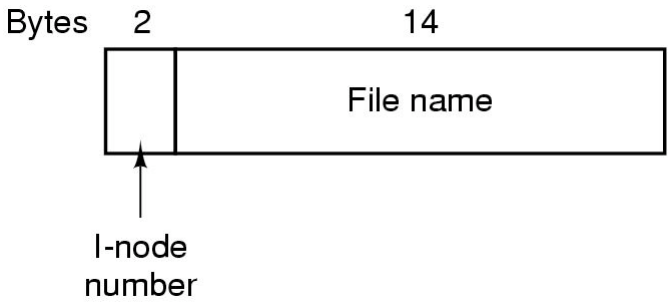
An entry for (part of) a long file name in Windows 98

The Windows 98 File System (3)

68	d o g	A	0	CK					0			
3	o v e	A	0	CK	t h e		l a		0	z y		
2	w n f o	A	0	CK	x	j u m p			0	s		
1	T h e q	A	0	CK	u i c k		b		0	r o		
T	H E Q U I ~ 1	A	NT	S	Creation time		Last acc	Upp	Last write	Low	Size	
Bytes												

An example of how a long name is stored in Windows 98

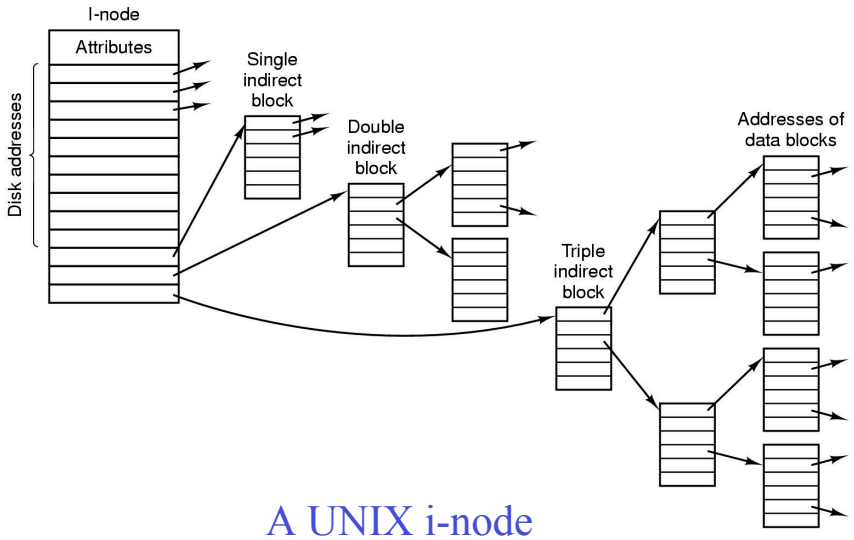
The UNIX V7 File System (1)



A UNIX V7 directory entry

438

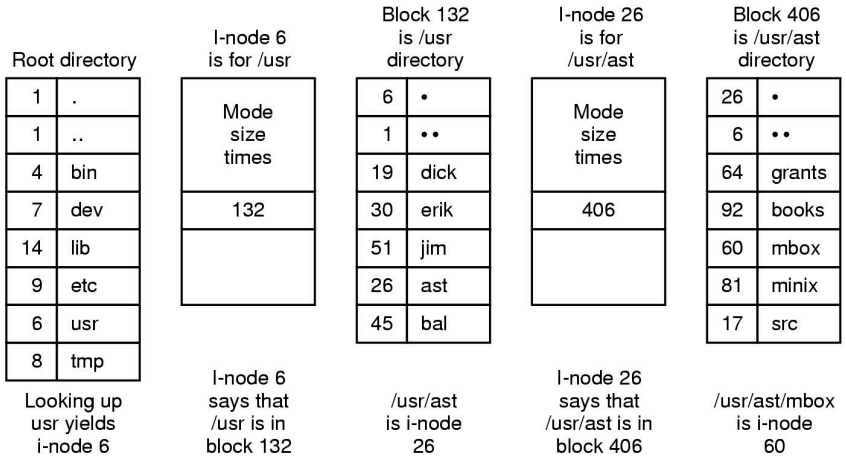
The UNIX V7 File System (2)



A UNIX i-node

439

The UNIX V7 File System (3)



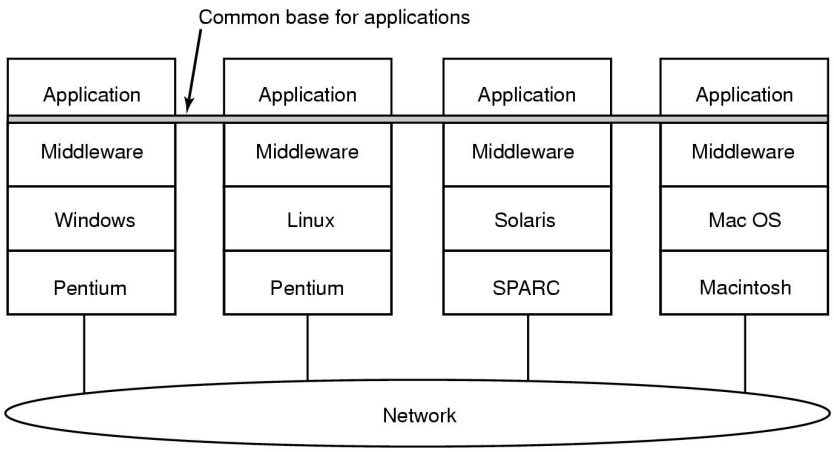
The steps in looking up */usr/ast/mbox*

Distributed Systems

Item	Multiprocessor	Multicomputer	Distributed System
Node configuration	CPU	CPU, RAM, net interface	Complete computer
Node peripherals	All shared	Shared exc. maybe disk	Full set per node
Location	Same rack	Same room	Possibly worldwide
Internode communication	Shared RAM	Dedicated interconnect	Traditional network
Operating systems	One, shared	Multiple, same	Possibly all different
File systems	One, shared	One, shared	Each node has own
Administration	One organization	One organization	Many organizations

Comparison of three kinds of multiple CPU systems

Distributed Systems: middleware



Achieving uniformity with middleware

Distributed File Systems

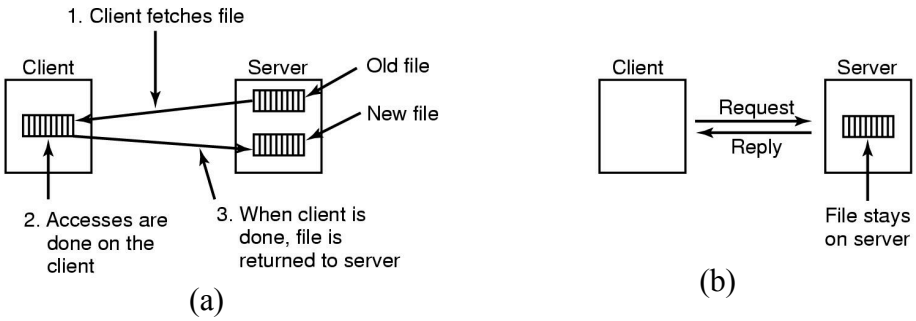
- networking computers, sharing files
- explicit file utilities: ftp, scp, etc.
  - sessions between pairs of computers

Distributed File System: single view of files

- easy access regardless of location
- sharing semantics
- file transfer modes
- directory structure

443

DFS transfer models



(a) upload/download model

(b) remote access model

DFS: directory structure

- provide directory, read, write, operations
  - but directory on more than one machine
- global v. local naming
  - is the view the same across all the DS?
- location transparency
  - is the location part of the file's name/path?
- location independence
  - if file moved, must its name change?

445

DFS: global v. local naming

users prefer to see the same everywhere, but:

- some files must be on specific computers
  - binaries; machine-state inits, etc.

thus some files must be localized visibly

- global dir structures: add a common root
  - retain meaning of "/" via ".." to access it
- local dir structures: mount remote on local dir
  - transparency but different views (same file has different paths on different computers)

446

DFS: global v. local naming (ctd)

Shared directory substructure (Andrew)

- 2-part hierarchy for each machine
  - local files, different for each machine
  - subtree of shared files, identical everywhere
- beneficial for performance
  - e.g., local temp files not shared

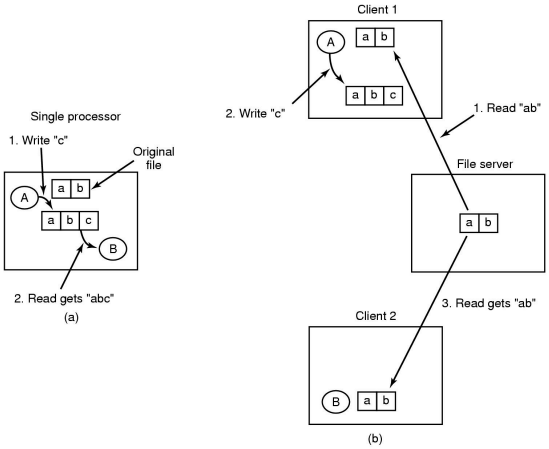
447

DFS: global v. local naming

(10-14)

448

DFS File Sharing: inconsistency



- Semantics of File sharing
  - (a) single processor gives sequential consistency
  - (b) distributed system may return obsolete value

DFS: semantics of file sharing

in centralized systems, unique access to file

- Unix semantics: writes visible immediately
  - read sees *last* write
  - writes propagated over network; order?
- session semantics: private copy until close
  - less overhead than Unix
  - less predictable: *lost update* problem
- transaction semantics (atomic set updates)
  - procs reduce/specify time changes invisible
- immutable file semantics (version mgmt)

450

DFS Implementation

- typically client-server architecture (NFS)
  - virtual FS layer hides local and NFS client
  - local system asks its NFS client for a file; client asks remote system's server for the file.
- overhead: network & disk access delay

451

DFS Implementation: Caching

overhead: network & disk access delay

- server caching: simple and transparent
  - caching entire file wasteful, complicated (why?)
  - cach already accessed blocks (LRU?)
- client caching, consistency problems
- write-through v. delayed writing
  - e.g., session semantics (propag to server at close)
- cache update schemes
  - other clients may have stale copies!
  - server-initiated (breaks C-S paradigm)
  - client-managed: how often check?

452

DFS Stateless v Stateful Servers

- new file = new entry in Open File Table
  - where is OFT maintained?
- stateful server maintains OFTs for clients
  - client simplified - only convey procs. reqs.
    - reliability: crashed client leaves useless OFTs; crashed server loses client OFTs
- stateless server: clients keep their OFTs
  - interactions more complicated
  - server can crash and recover -idempotency-

453

DFS: Sun's NFS

- clients access files through set of RPCs
  - *most* operations are **idempotent**
    - the ones that aren't generate error messages, e.g., rename/delete if not ack'd is repeated
- caching at server *and* clients, but also Unix semantics, a conflict
  - client caching for nonshared files only
  - thus server needs to keep some state (!)
- mixed semantics, but popular/useful
    - also see Castro & Liskov BASE work

454

DFS Stateless v Stateful Servers

- (10-16)

455

DFS: File Replication

multiple copies of a file on multiple servers

- availability: wrt to crashes
- reliability: can reconstruct consistent state
- performance: find a nearby copy
- scalability: deal well with load

replication is a form of caching; again the problem is *consistency*

456

DFS: File Replication 2

Protocols to keep replicas consistent:

- Read-Any/Write-All
  - W-A: propagate write w/out interleaving with other updates (consistency) - multicast
    - sometimes apps allow interleaved updates
    - what if one copy is unavailable?

457

DFS: File Replication 3

Protocols to keep replicas consistent:

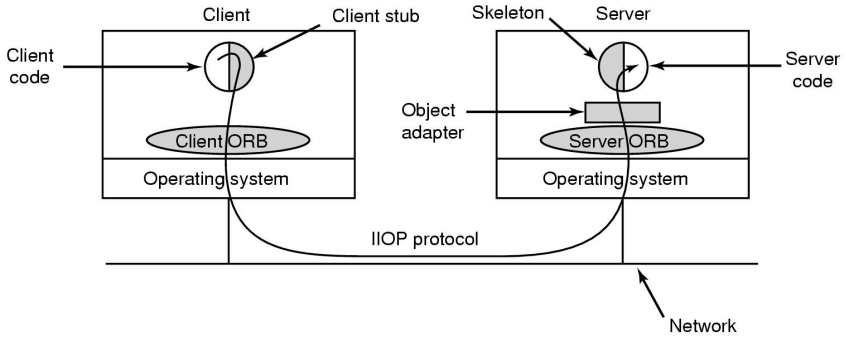
- Quorum-based R/W protocol on *N* replicas
  - *w:* write quorum, # replicas must be written
  - *r:* read quorum, # replicas must be read (!)

Then we require:

- $r + w > N$ (any read intersects W quorum)
- $w > N/2$ (no disjoint subsets updated)

458

Object-level "DFS:" CORBA



Main elements of CORBA based system

- Common Object Request Broker Architecture