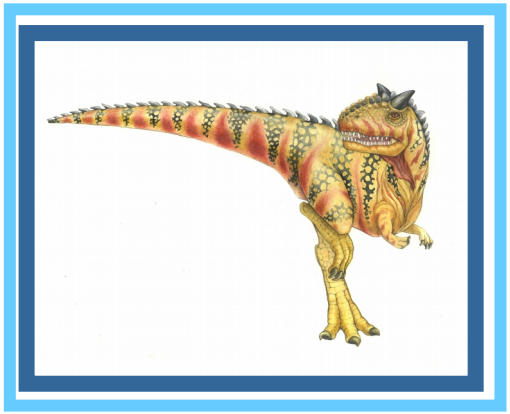
Chapter 11: File System Implementation



Chapter 11: File System Implementation

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

Objectives

- To describe the details of implementing local file systems and directory structures
- To describe the implementation of remote file systems
- To discuss block allocation and free-block algorithms and trade-offs

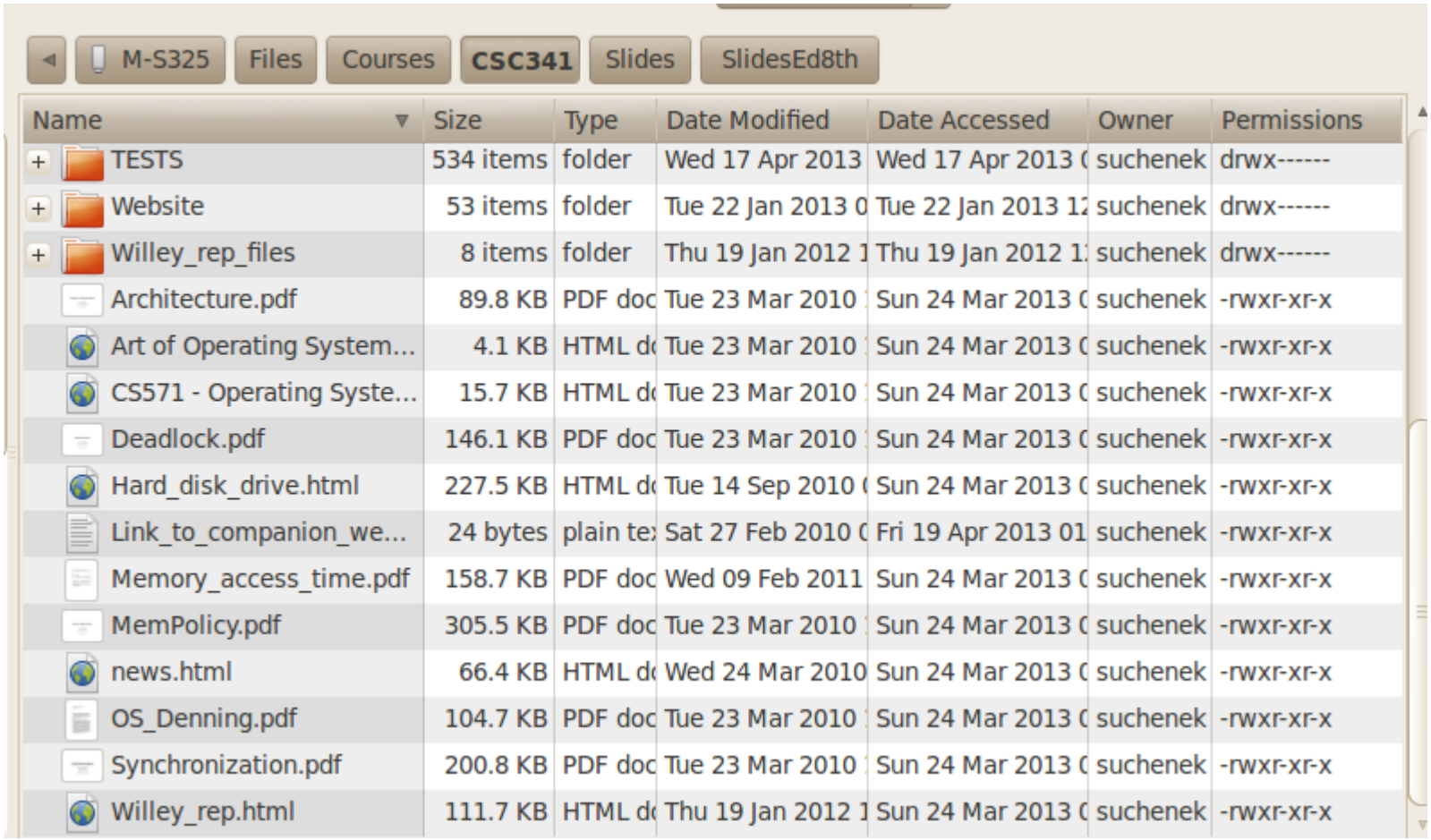
File-System Structure

- File structure
  - Logical storage unit: logical block
    - **NOTE**: The smallest-addressable unit of information is block (a.k.a. physical block) which for a hard disk the textbook refers to as a sector (in the meaning of track sector). Thus logical block is a sequence of physical blocks.
  - Collection of related information
- File system organized into layers
- File system resides on secondary storage (disks)
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- File control block – storage structure consisting of information about a file
- Device driver controls the physical device

A Typical File Control Block

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

A Typical File Control Block



Name	Size	Type	Date Modified	Date Accessed	Owner	Permissions
TESTS	534 items	folder	Wed 17 Apr 2013	Wed 17 Apr 2013 (	suchenek	drwx------
Website	53 items	folder	Tue 22 Jan 2013 0	Tue 22 Jan 2013 12	suchenek	drwx------
Willey_rep_files	8 items	folder	Thu 19 Jan 2012 1	Thu 19 Jan 2012 1	suchenek	drwx------
Architecture.pdf	89.8 KB	PDF doc	Tue 23 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
Art of Operating System...	4.1 KB	HTML d	Tue 23 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
CS571 - Operating Syste...	15.7 KB	HTML d	Tue 23 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
Deadlock.pdf	146.1 KB	PDF doc	Tue 23 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
Hard_disk_drive.html	227.5 KB	HTML d	Tue 14 Sep 2010 (	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
Link_to_companion_we...	24 bytes	plain te	Sat 27 Feb 2010 (	Fri 19 Apr 2013 01	suchenek	-rwxr-xr-x
Memory_access_time.pdf	158.7 KB	PDF doc	Wed 09 Feb 2011	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
MemPolicy.pdf	305.5 KB	PDF doc	Tue 23 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
news.html	66.4 KB	HTML d	Wed 24 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
OS_Denning.pdf	104.7 KB	PDF doc	Tue 23 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
Synchronization.pdf	200.8 KB	PDF doc	Tue 23 Mar 2010	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x
Willey_rep.html	111.7 KB	HTML d	Thu 19 Jan 2012 1	Sun 24 Mar 2013 (	suchenek	-rwxr-xr-x

Directory Implementation

- **Linear list** of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute
- **Hash Table** – linear list with hash data structure.
  - decreases directory search time
  - **collisions** – situations where two file names hash to the same location
  - fixed size

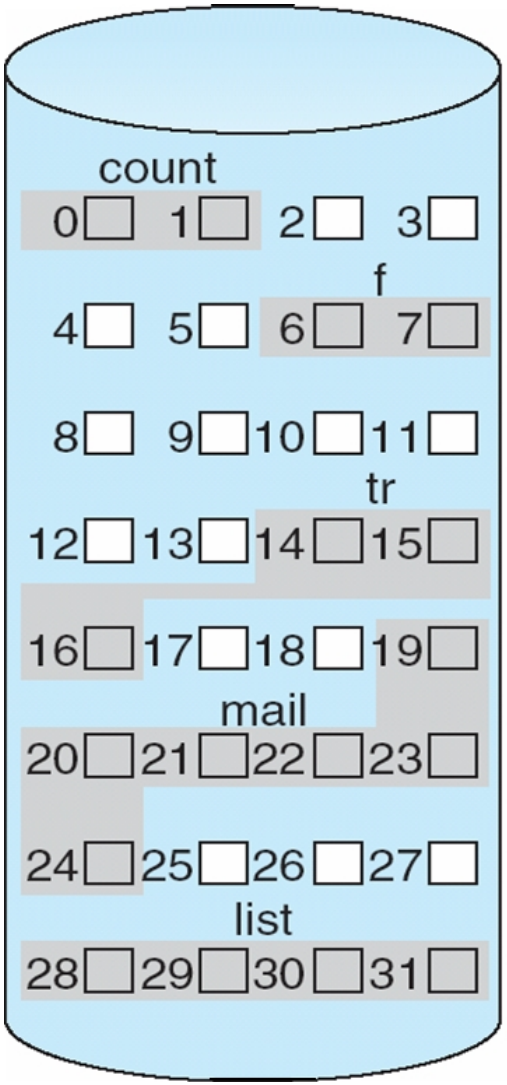
Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
- Contiguous allocation
- Linked allocation
- Indexed allocation

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Random access
- Fast transfer – work well with DMA
- Potentially wasteful of space – may cause external fragmentation problem.
- Files cannot grow unrestrictedly

Contiguous Allocation of Disk Space



directory

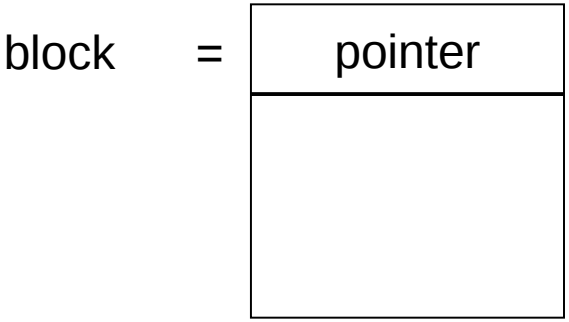
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked Allocation

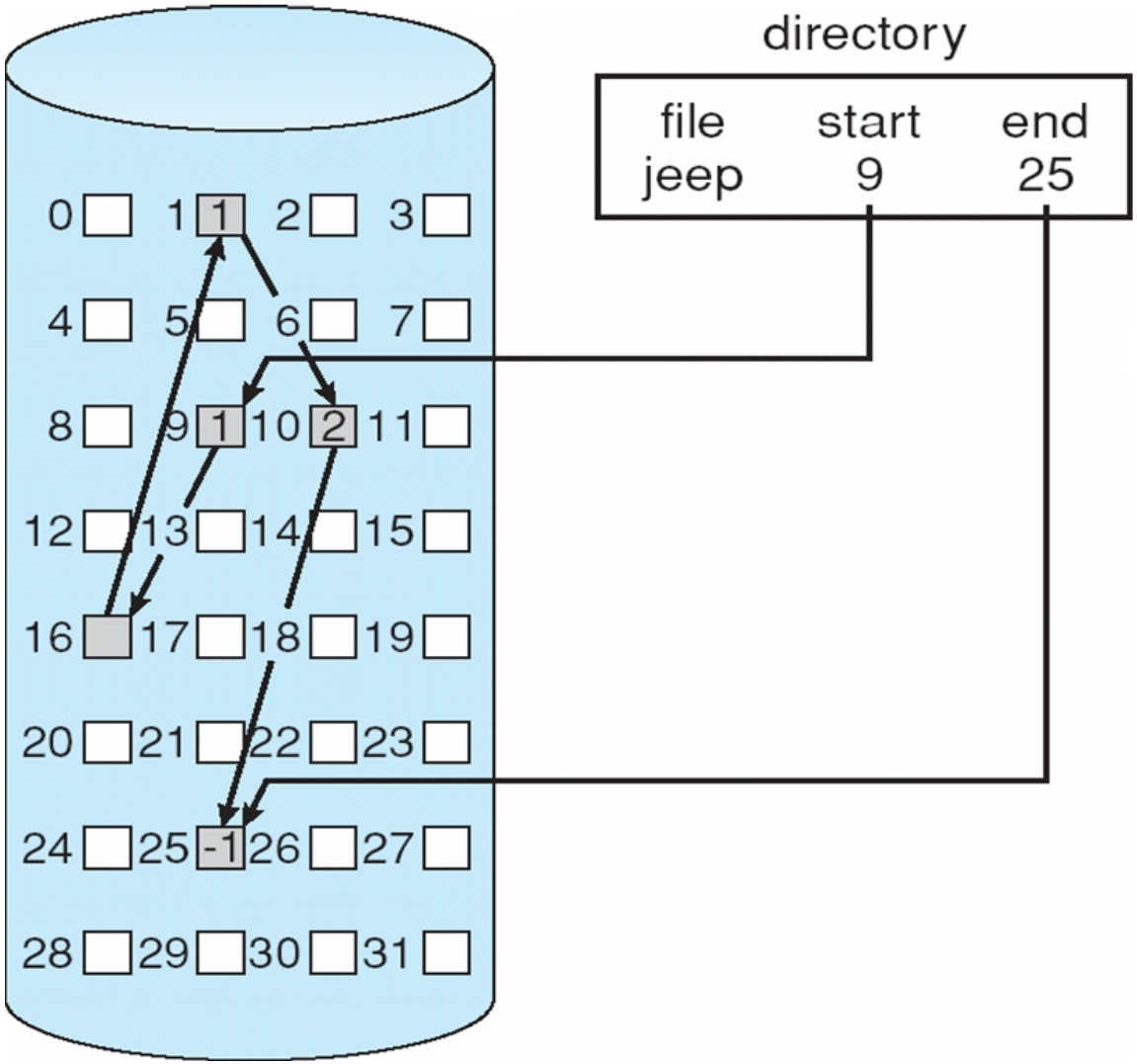
- Each file occupies a “linked list” of blocks on the disk
- Versatile and relatively simple
- Sequential access – Θ(N)
- Not so fast transfer – may require many calls to DMA
- Good utilization of space – no external fragmentation problem.
- Files can grow unrestrictedly
- Relatively unreliable and prone to failures

Linked Allocation

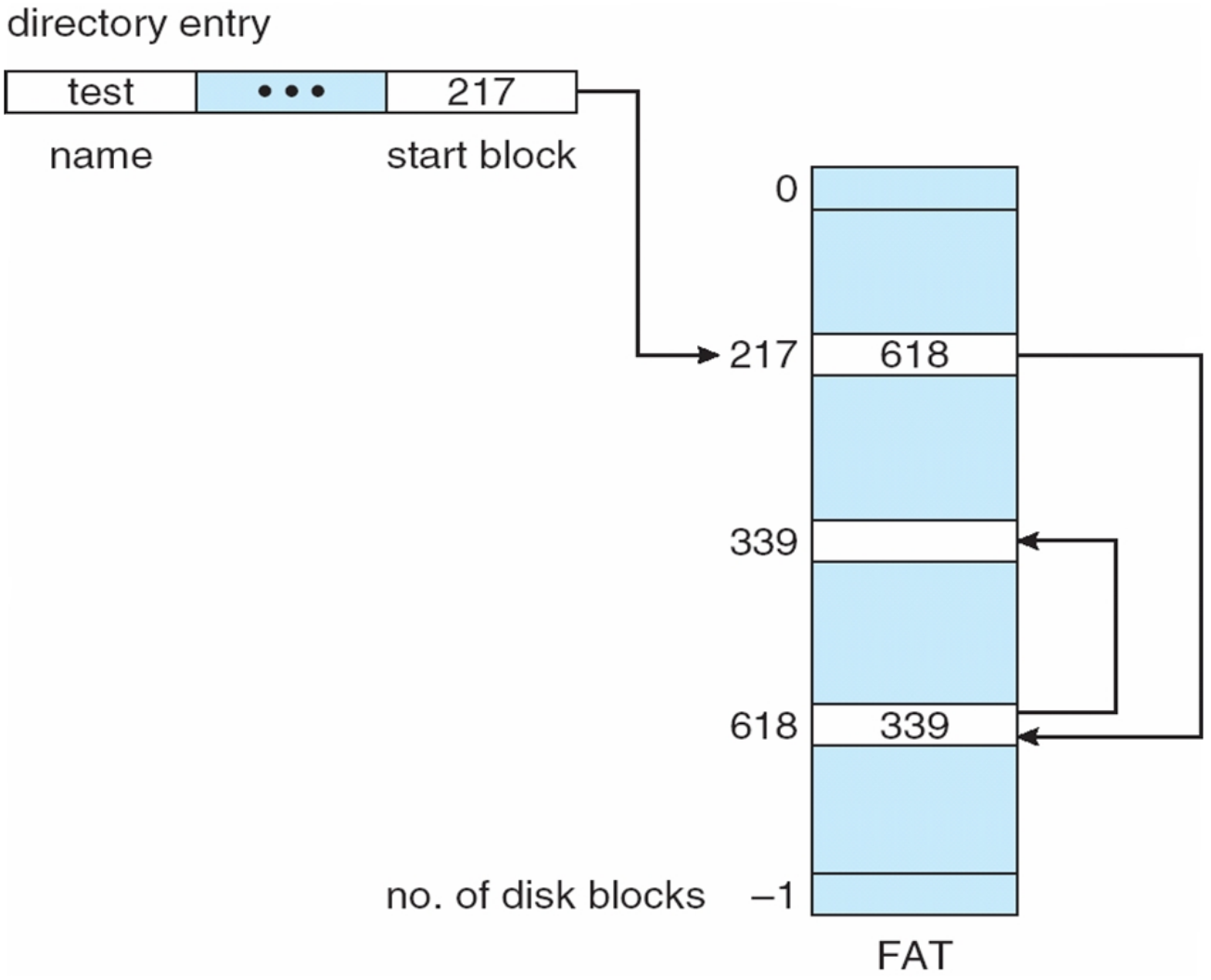
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.



Linked Allocation



File-Allocation Table

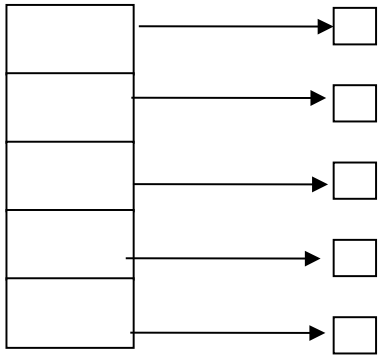


Indexed Allocation

- Each file is represented as a B-tree of blocks on the disk
- Allows fast access to blocks in large files
- Semi-random access – Θ(log N)
- Not so fast transfer – may require many calls to DMA
- Good utilization of space – no external fragmentation problem.
- Files can grow unrestrictedly
- Extra space needed to implement the B-tree structure

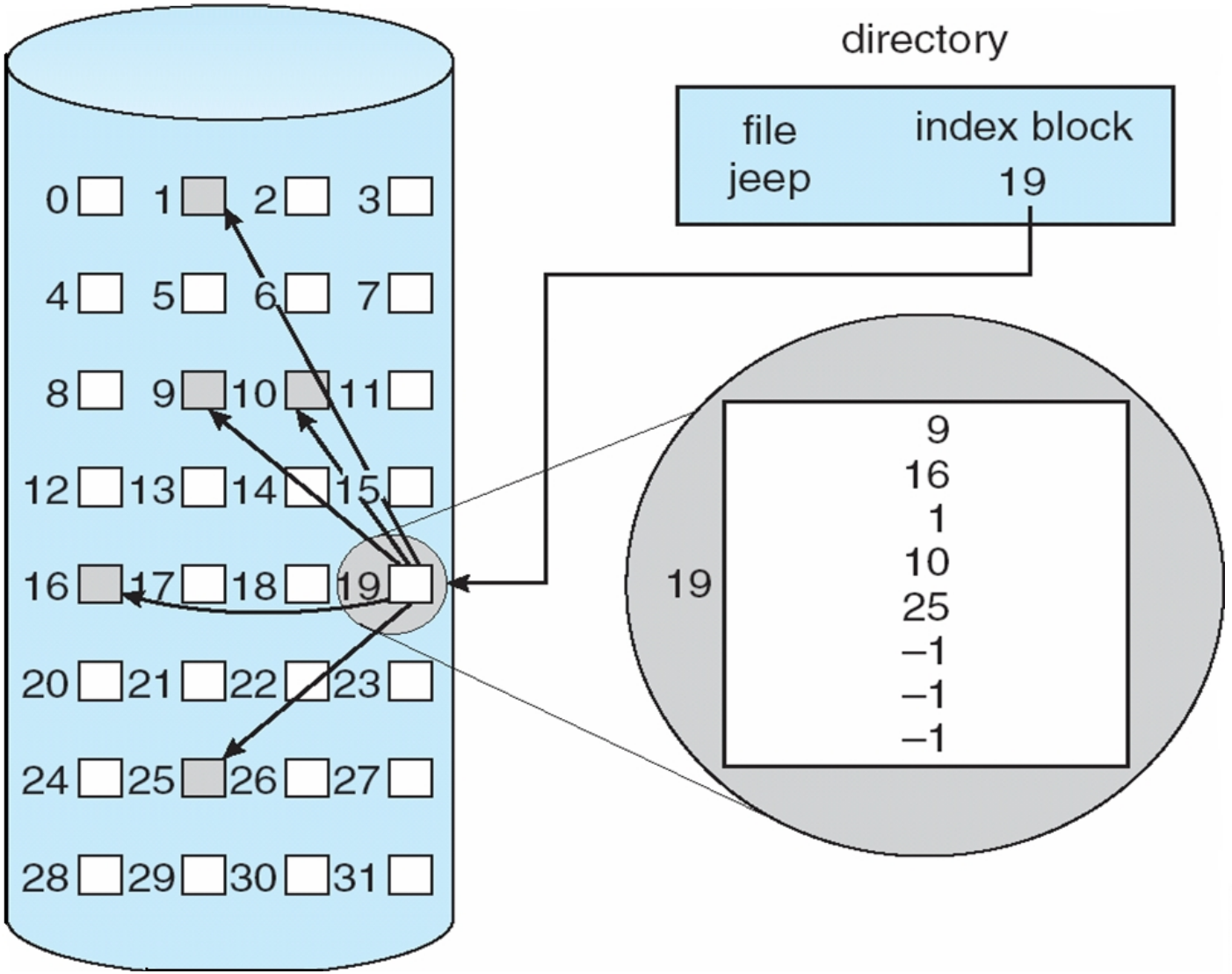
Indexed Allocation

- Brings all pointers together into the index block
- Logical view

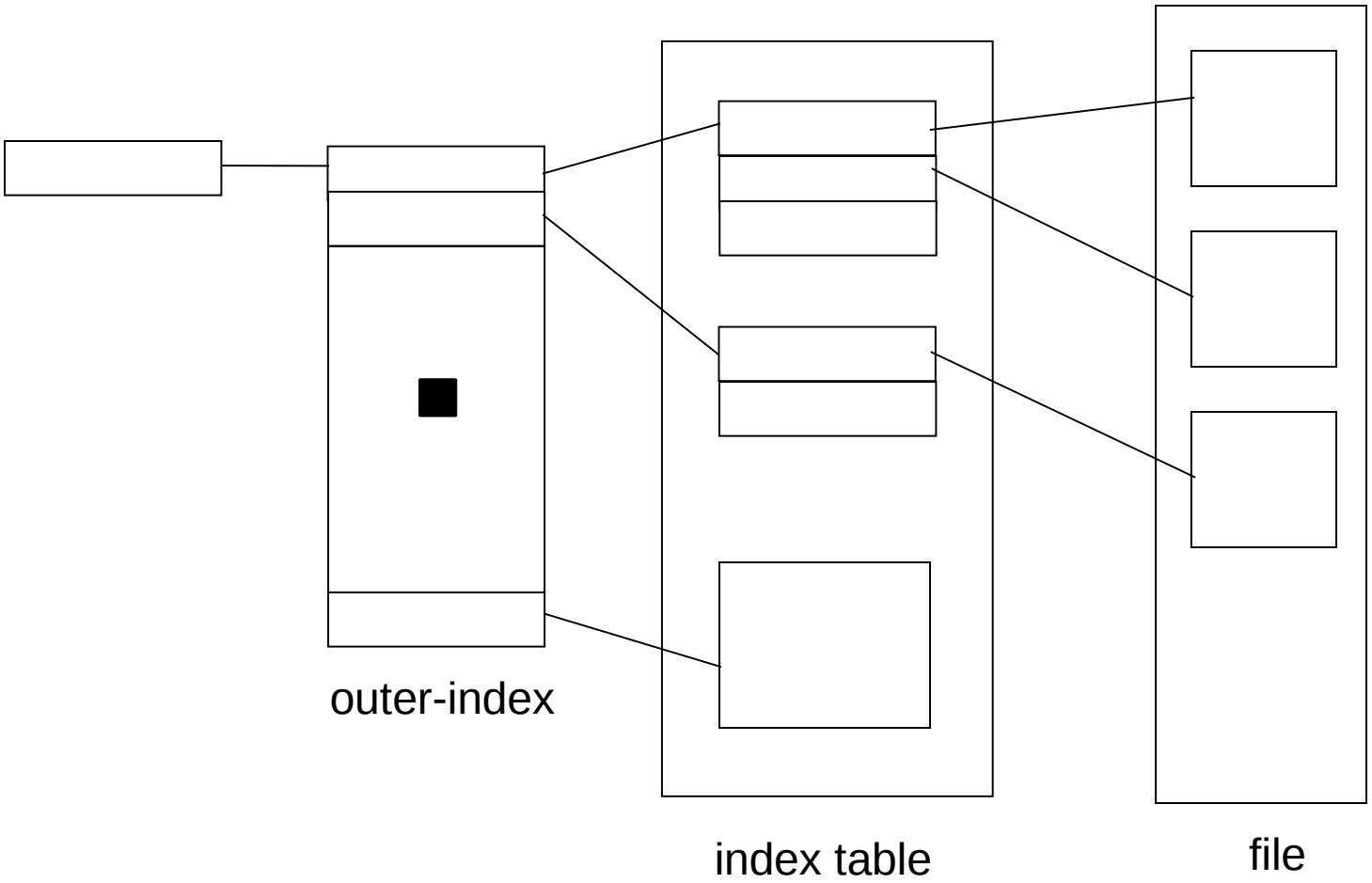


index table

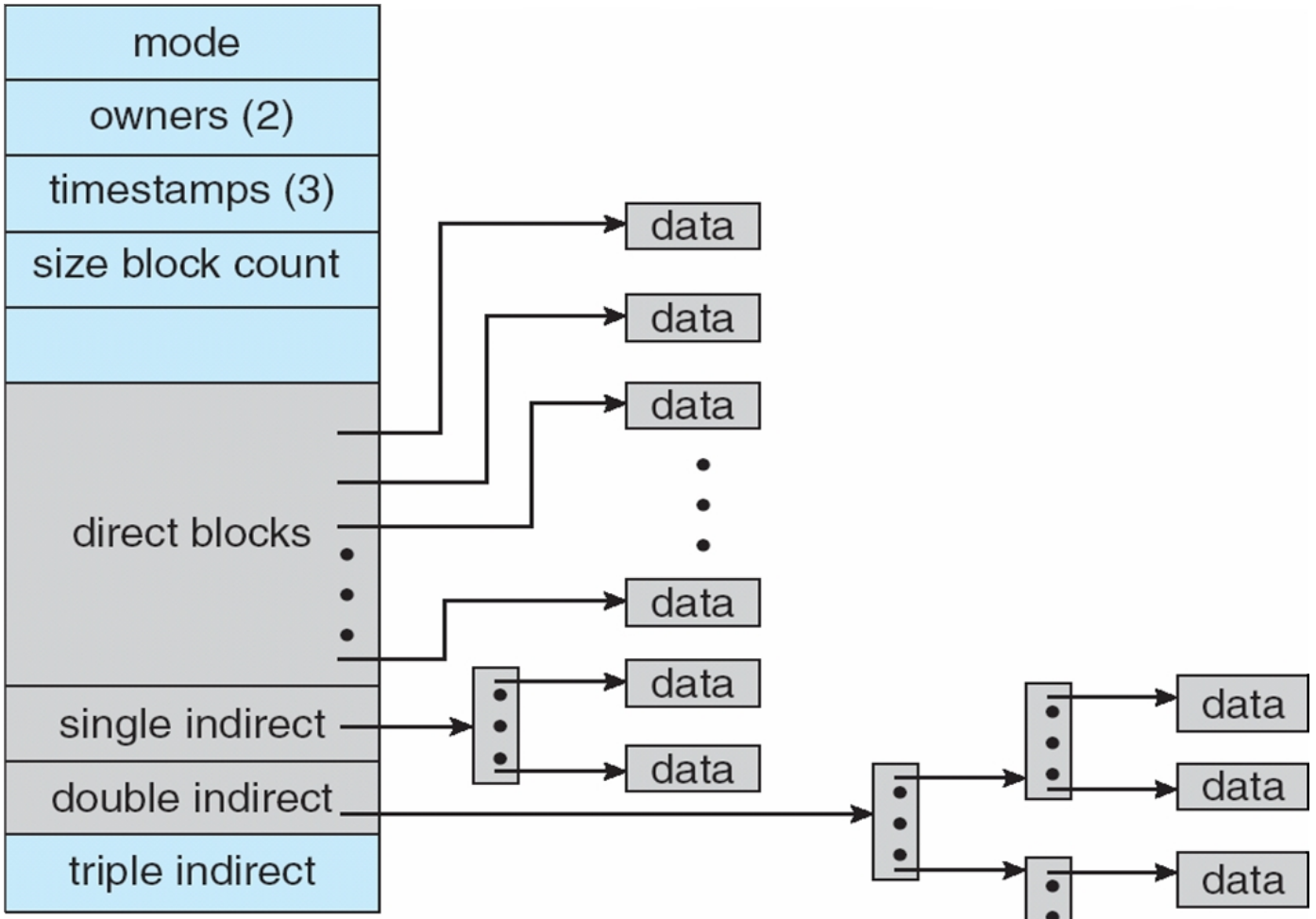
Example of Indexed Allocation



Indexed Allocation – Mapping (Cont.)

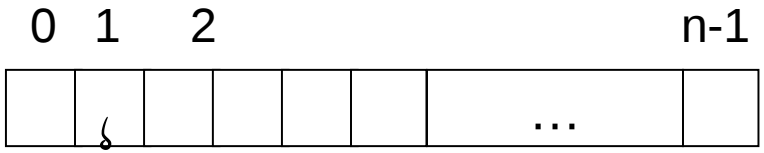


Combined Scheme: UNIX UFS (4K bytes per block)



Free-Space Management

- Bit vector (*n* blocks)



bit[*i*] = 0 ⇒ block[*i*] free

1 ⇒ block[*i*] occupied

Block number calculation

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

Free-Space Management (Cont.)

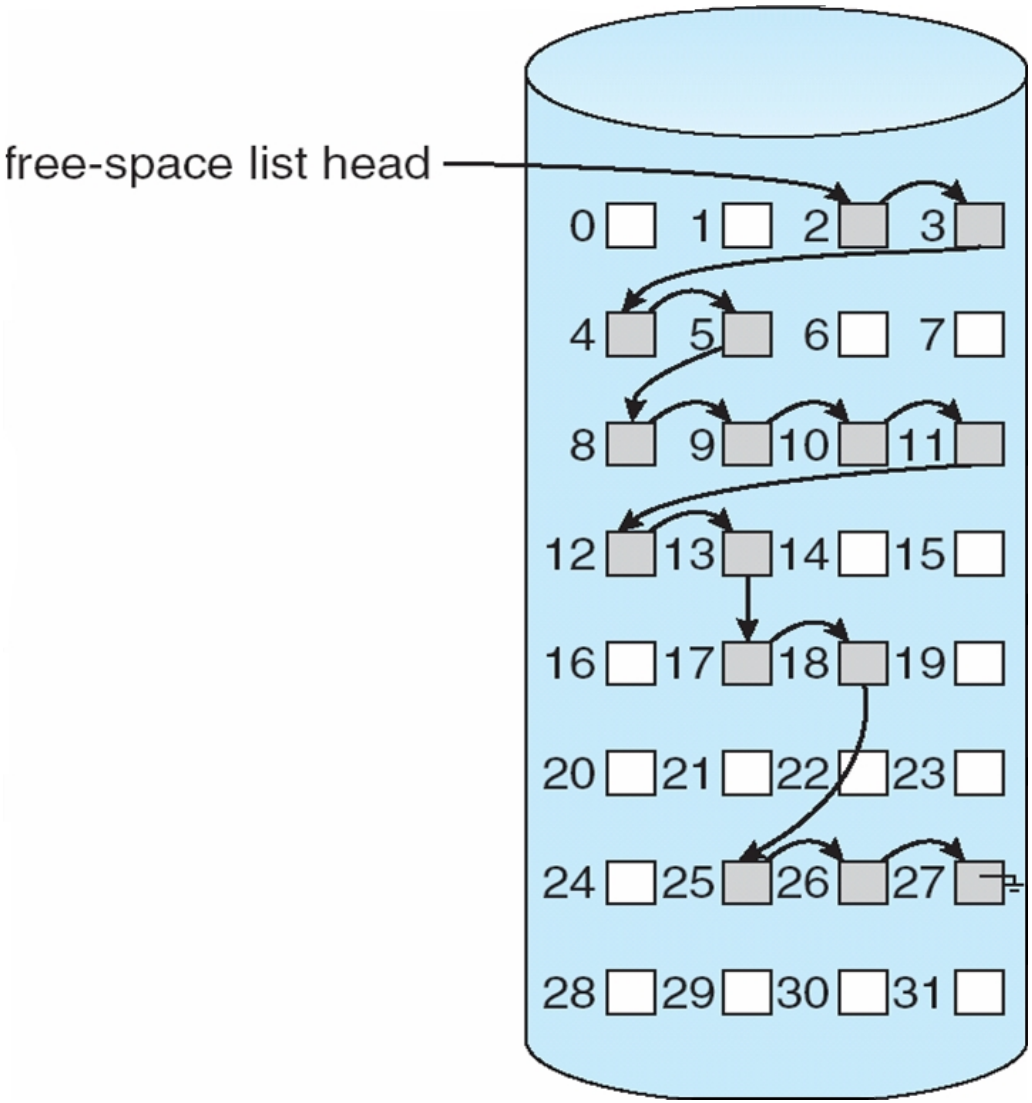
- Bit map requires extra space
  - Example:

    block size = 2^{12} bytes

    disk size = 2^{30} bytes (1 gigabyte)

    $n = 2^{30}/2^{12} = 2^{18}$ bits = 2^{15} bytes = 32 kilobytes
- Easy to get contiguous files
- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
- Grouping
- Counting

Linked Free Space List on Disk



Free-Space Management (Cont.)

- Grouping
  - Modify linked list to store address of next *n-1* free blocks in first free block, plus a pointer to next block that contains free-block-pointers (like this one)
- Counting
  - Because space is frequently contiguously used and freed, with contiguous-allocation allocation, extents, or clustering
    - Keep address of first free block and count of following free blocks
    - Free space list then has entries containing addresses and counts

Efficiency and Performance

- Efficiency dependent on:
  - disk allocation and directory algorithms
  - types of data kept in file's directory entry
- Performance
  - disk cache – separate section of main memory for frequently used blocks
  - free-behind and read-ahead – techniques to optimize sequential access
  - improve PC performance by dedicating section of memory as virtual disk, or RAM disk

The Sun Network File System (NFS)

The remainder of this deck is optional for all students

- An implementation and a specification of a software system for accessing remote files across LANs (or WANs)
- The implementation is part of the Solaris and SunOS operating systems running on Sun workstations using an unreliable datagram protocol (UDP/IP protocol and Ethernet

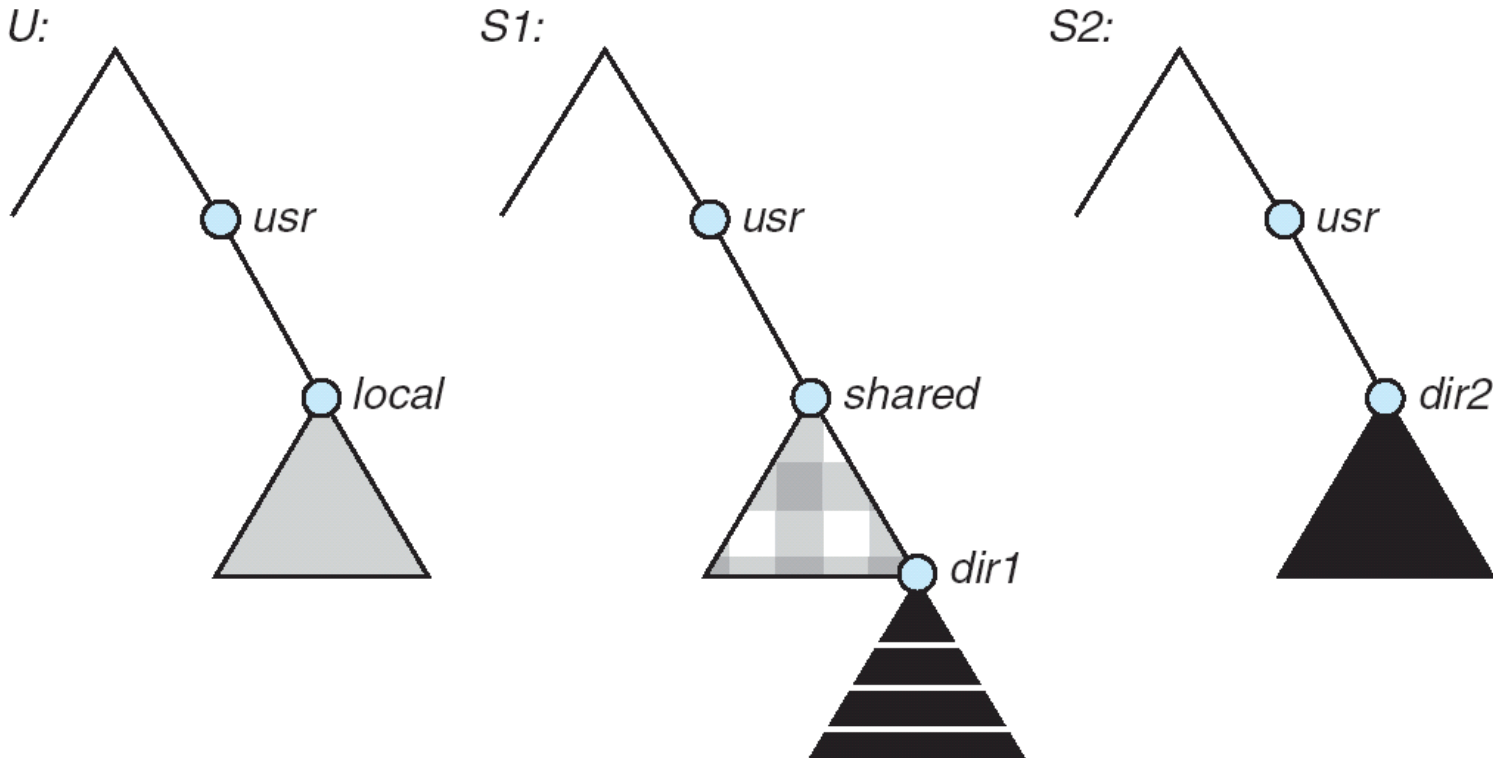
NFS (Cont.)

- Interconnected workstations viewed as a set of independent machines with independent file systems, which allows sharing among these file systems in a transparent manner
  - A remote directory is mounted over a local file system directory
    - The mounted directory looks like an integral subtree of the local file system, replacing the subtree descending from the local directory
  - Specification of the remote directory for the mount operation is nontransparent; the host name of the remote directory has to be provided
    - Files in the remote directory can then be accessed in a transparent manner
  - Subject to access-rights accreditation, potentially any file system (or directory within a file system), can be mounted remotely on top of any local directory

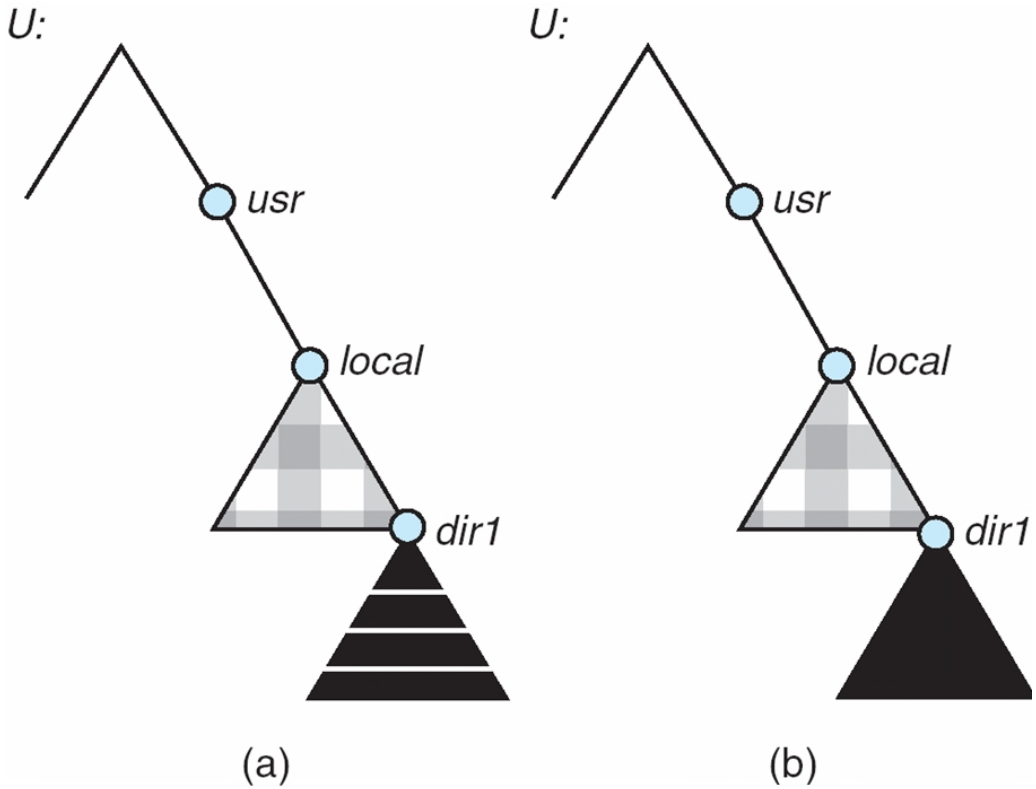
NFS (Cont.)

- NFS is designed to operate in a heterogeneous environment of different machines, operating systems, and network architectures; the NFS specifications independent of these media
- This independence is achieved through the use of RPC primitives built on top of an External Data Representation (XDR) protocol used between two implementation-independent interfaces
- The NFS specification distinguishes between the services provided by a mount mechanism and the actual remote-file-access services

Three Independent File Systems



Mounting in NFS



Mounts

Cascading mounts

NFS Mount Protocol

- Establishes 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 for mounting, 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
- File handle – a file-system identifier, and an inode number to identify the mounted directory within the exported file system
- The mount operation changes only the user's view and does not affect the server side

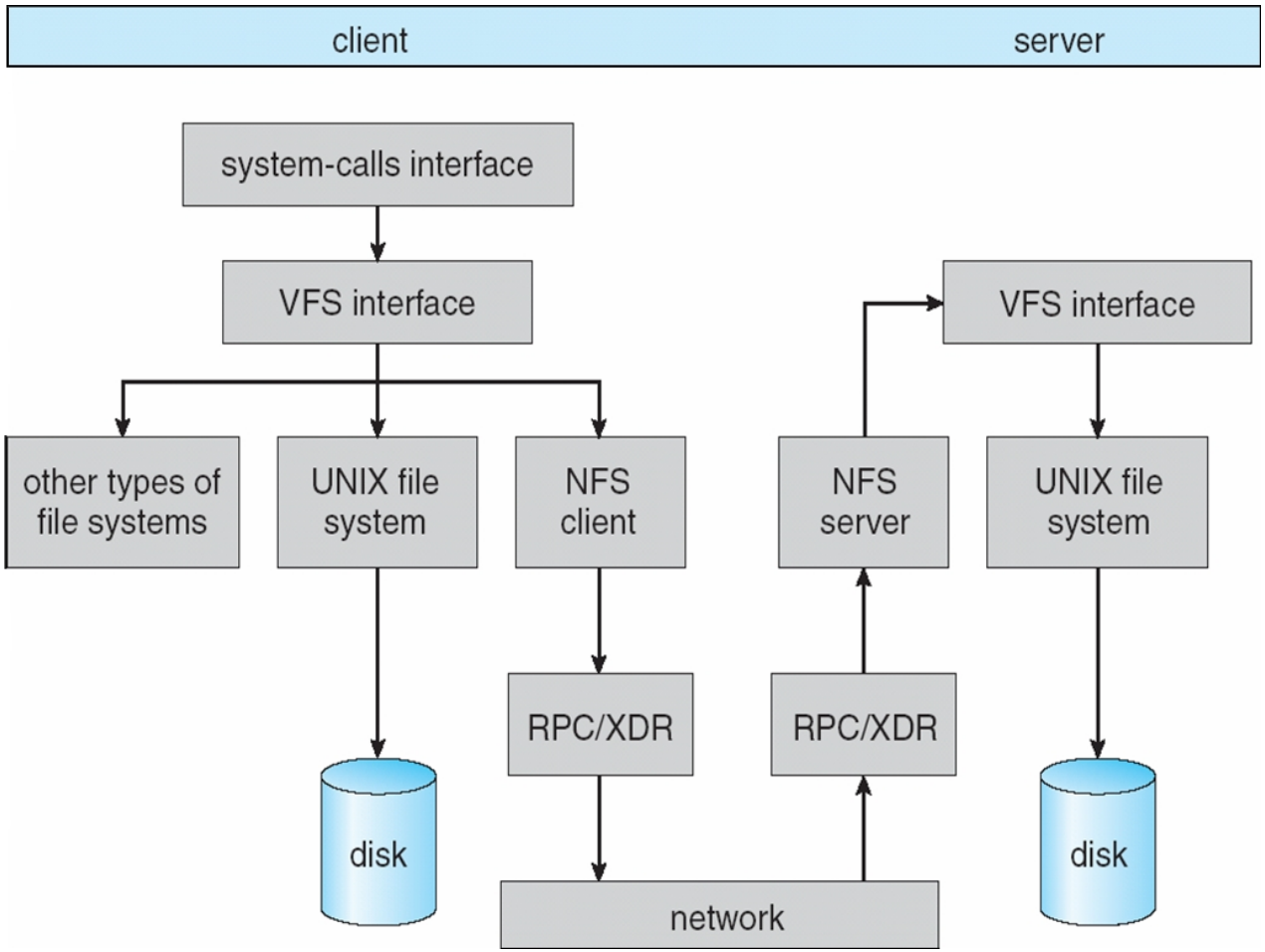
NFS Protocol

- 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 (NFS V4 is just coming available – very different, stateful)
- Modified data must be committed to the server's disk before results are returned to the client (lose advantages of caching)
- The NFS protocol does not provide concurrency-control mechanisms

Three Major Layers of NFS Architecture

- UNIX file-system interface (based on the **open, read, write**, and **close** calls, and **file descriptors**)
- Virtual File System (VFS) layer – distinguishes local files from remote ones, and local files are further distinguished according to their file-system types
  - The VFS activates file-system-specific operations to handle local requests according to their file-system types
  - Calls the NFS protocol procedures for remote requests
- NFS service layer – bottom layer of the architecture
  - Implements the NFS protocol

Schematic View of NFS Architecture



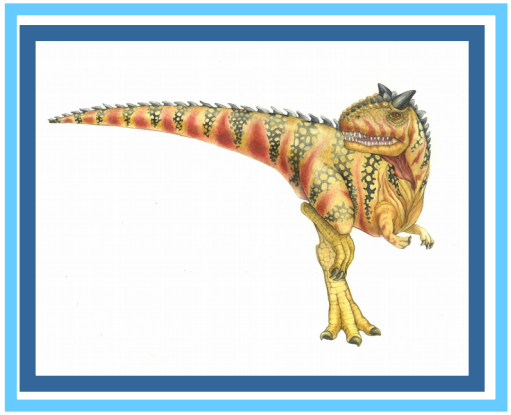
NFS Path-Name Translation

- Performed by breaking the path into component names and performing a separate NFS lookup call for every pair of component name and directory vnode
- To make lookup faster, a directory name lookup cache on the client's side holds the vnodes for remote directory names

NFS Remote Operations

- Nearly one-to-one correspondence between regular UNIX system calls and the NFS protocol RPCs (except opening and closing files)
- NFS adheres to the remote-service paradigm, but employs buffering and caching techniques for the sake of performance
- File-blocks cache – when a file is opened, the kernel checks with the remote server whether to fetch or revalidate the cached attributes
  - Cached file blocks are used only if the corresponding cached attributes are up to date
- File-attribute cache – the attribute cache is updated whenever new attributes arrive from the server
- Clients do not free delayed-write blocks until the server confirms that the data have been written to disk

End of Chapter 11



Now you know

Operating Systems