Chap. 14) File System Implementation

Chap. 15) File System Internals

경희대학교 컴퓨터공학과

조 진 성

Overview

User's view on file systems:

- ✓ How files are named?
- ✓ What operations are allowed on them?
- ✓ What the directory tree looks like?

Implementer's view on file systems:

- ✓ How files and directories are stored?
- ✓ How disk space is managed?
- ✓ How to make everything work efficiently and reliably?

File System Implementation

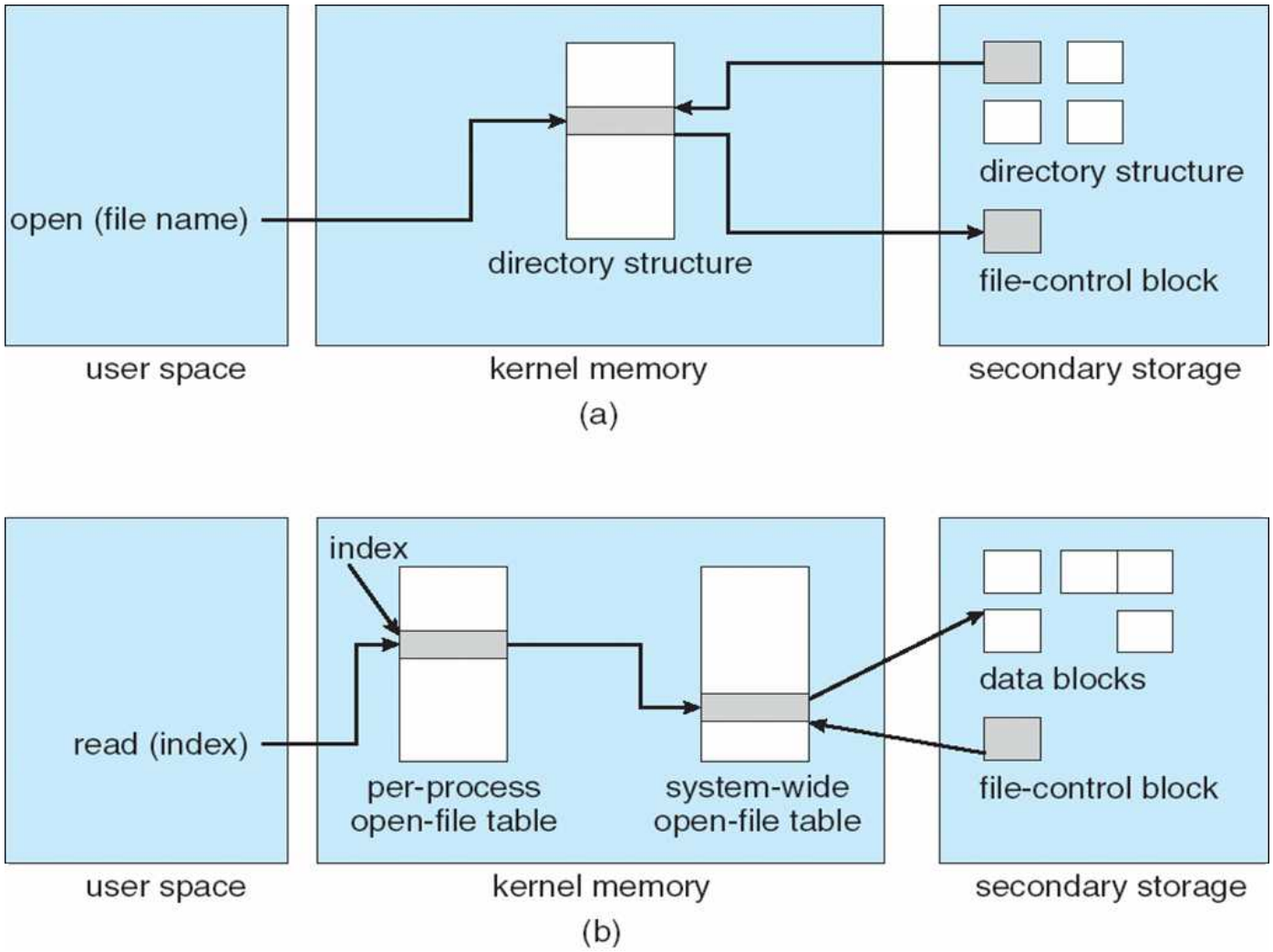
In-memory structure

- ✓ In-memory partition table
- ✓ In-memory directory structure
- ✓ System-wide open file table
- ✓ Per-process open file table

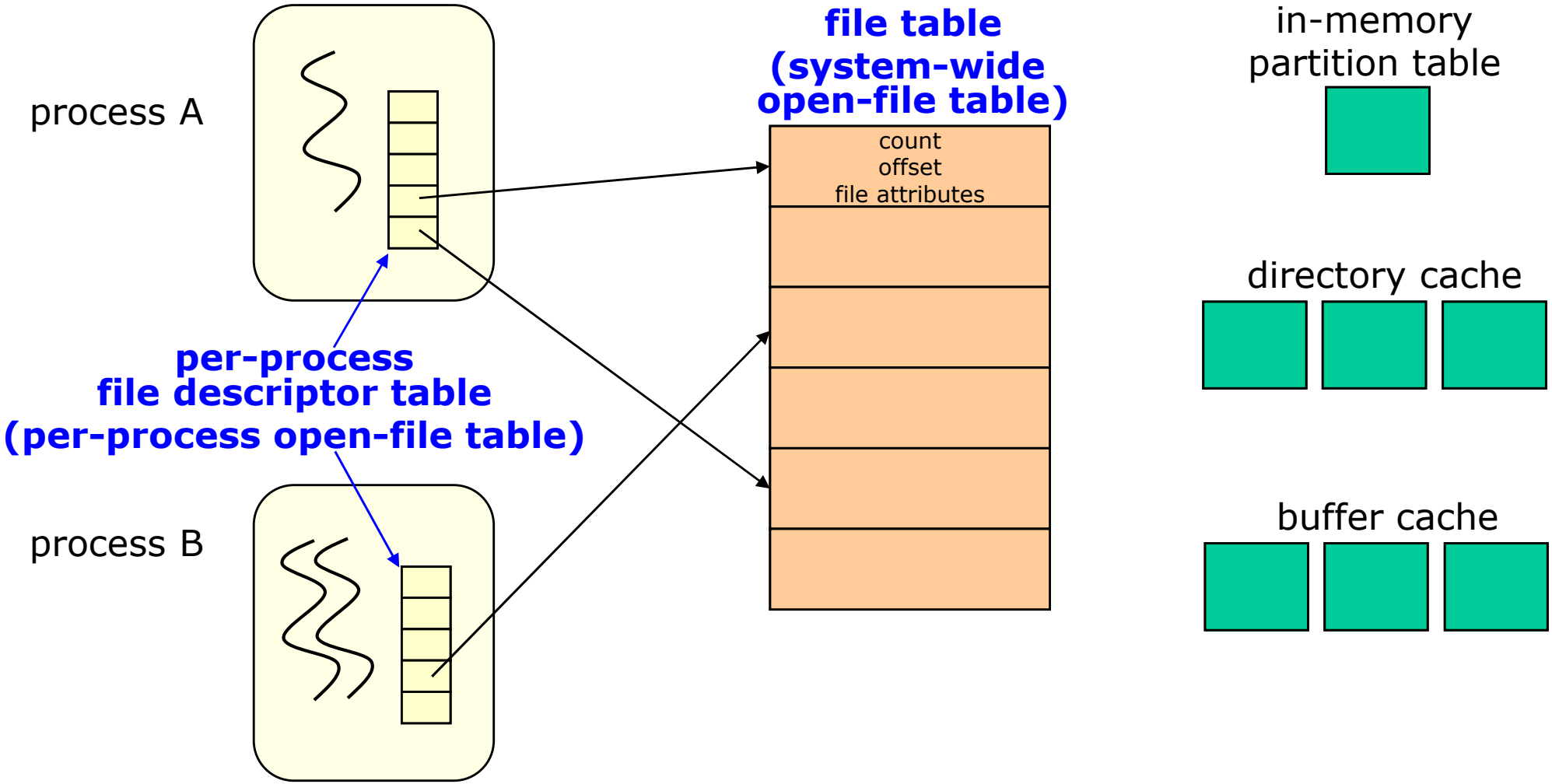
On-disk structure

- ✓ Boot control block
  - Boot block(UFS) or Boot sector(NTFS)
- ✓ Volume control block
  - Super block(UFS) or Master file table(NTFS)
- ✓ Directory structure
- ✓ File control block (FCB)
  - i-node(UFS) or in master file table(NTFS)

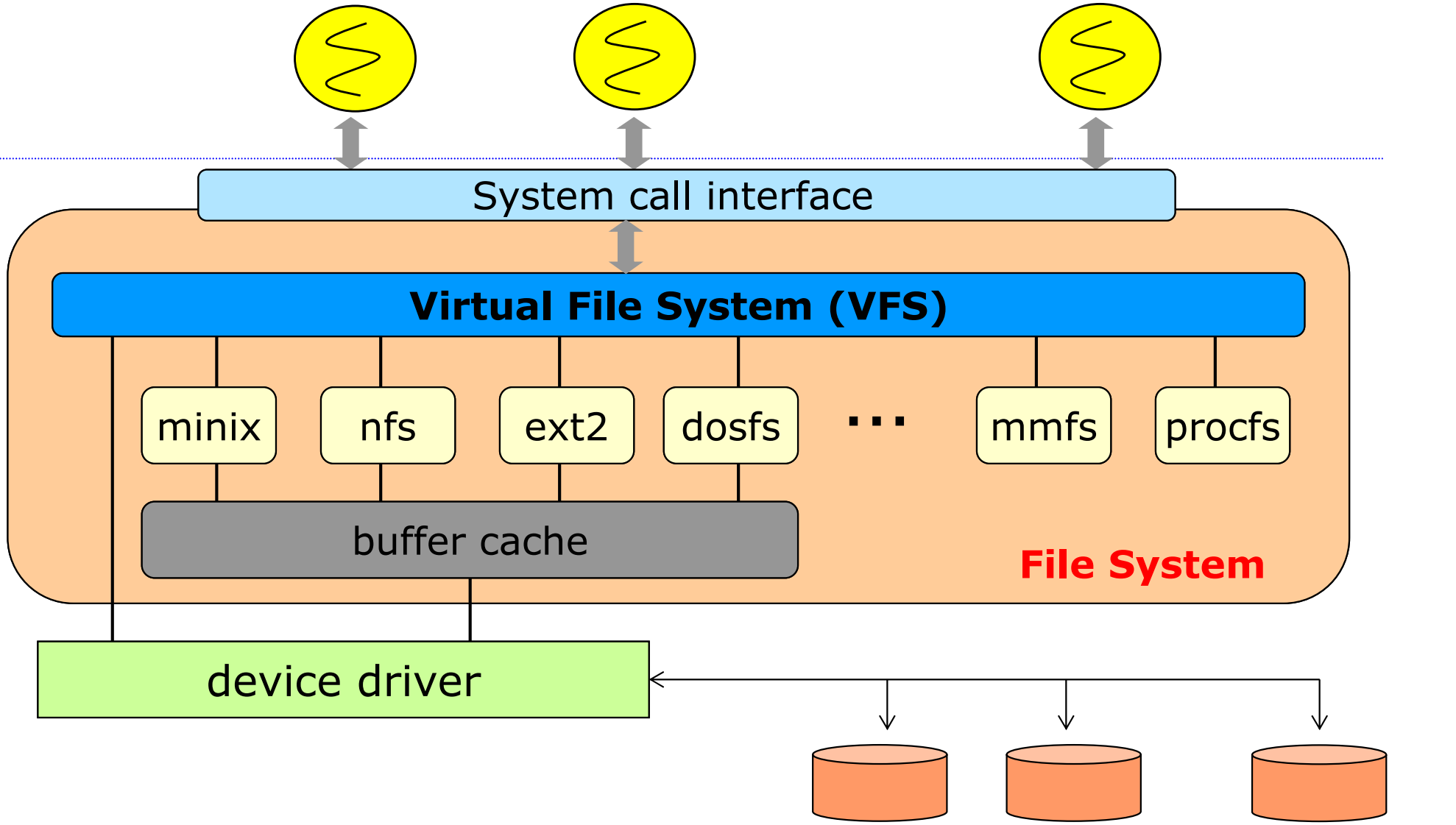
In-Memory Structure



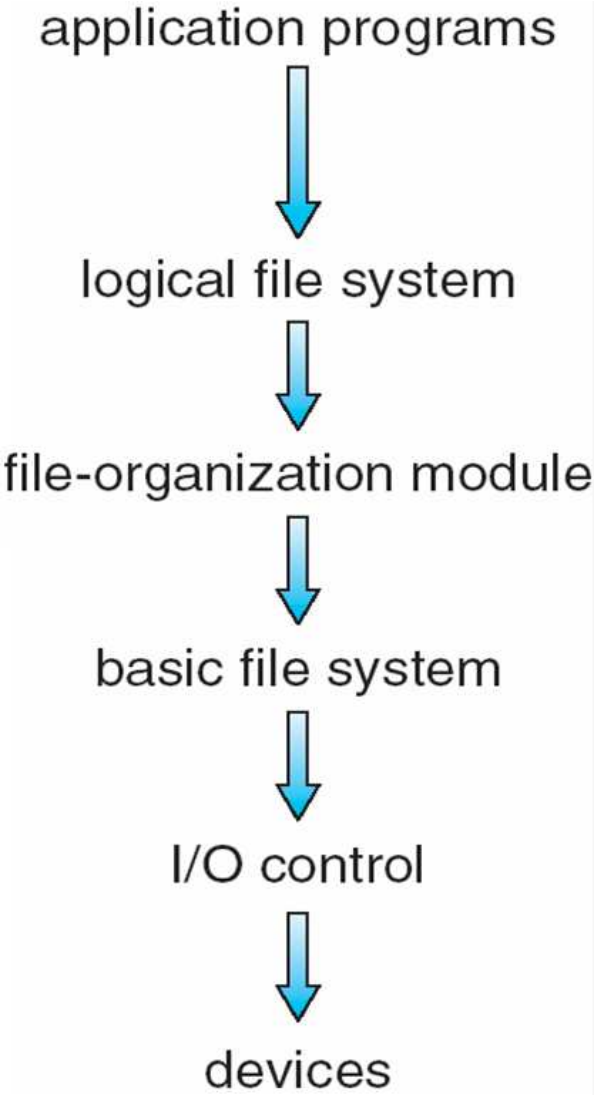
In-Memory Structure



Virtual File System



Layered File System



CD-ROM

- ✓ ISO 9660

Unix

- ✓ UFS, FFS

Windows

- ✓ FAT, FAT32, NTFS

Linux

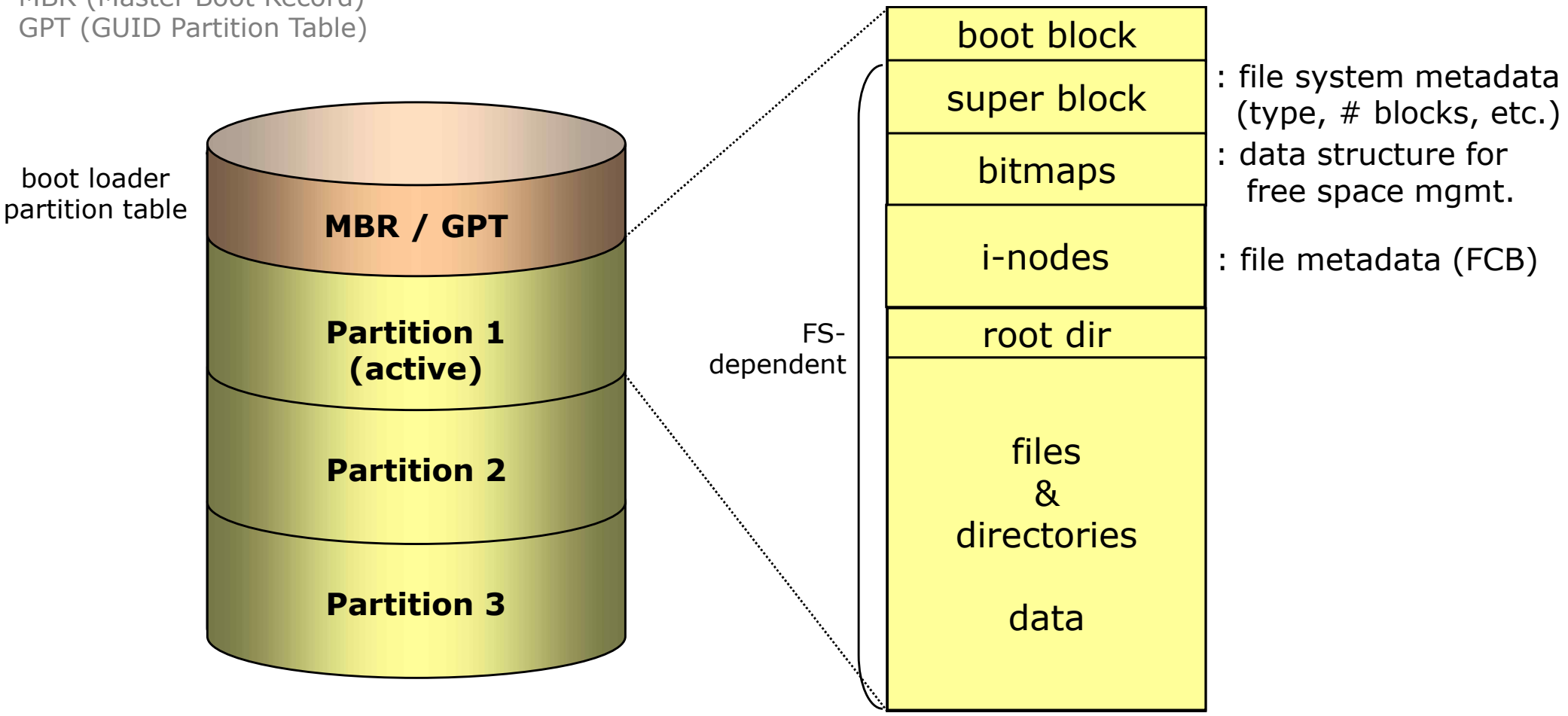
- ✓ Over 130 different file systems
- ✓ Extended file system, ext2/3/4
- ✓ Distributed file system

New ones

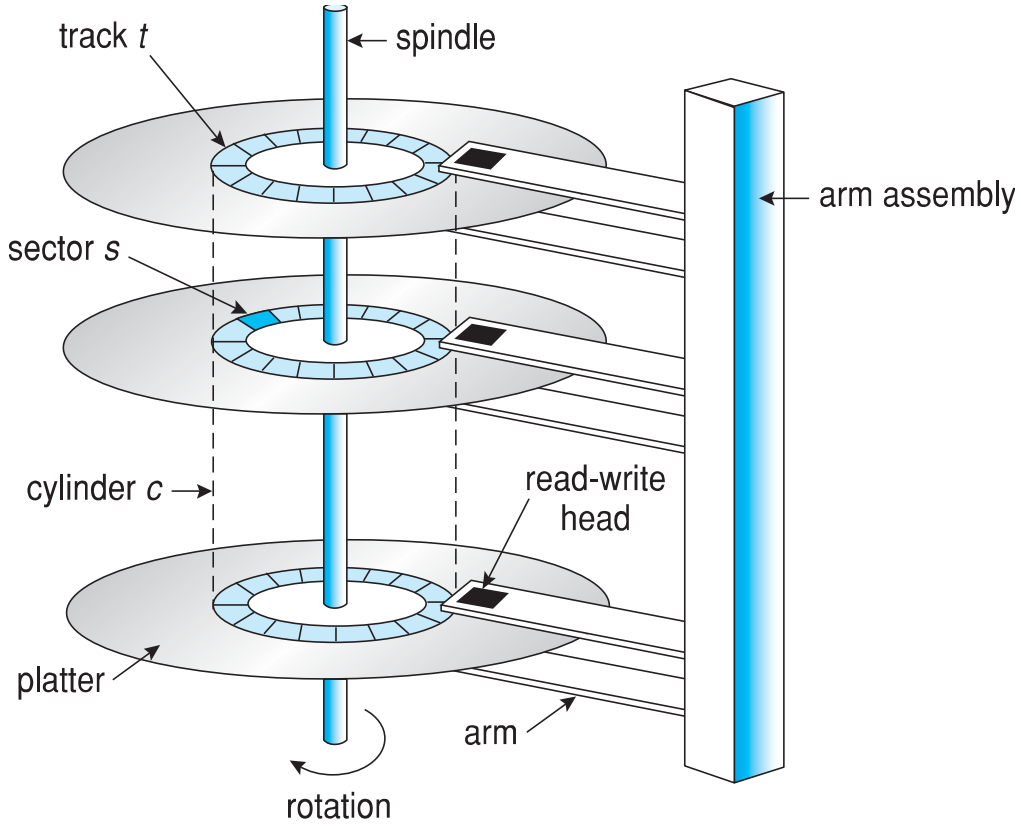
- ✓ ZFS, GoogleFS, Oracle ASM, FUSE, …

On-Disk Structure

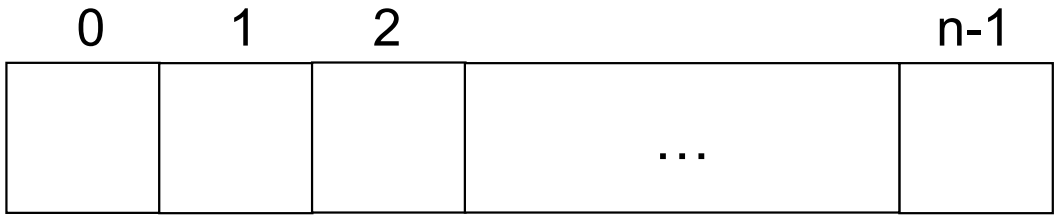
BIOS (Basic Input/Output System)
UEFI (Unified Extensible Firmware Interface)
MBR (Master Boot Record)
GPT (GUID Partition Table)



Disk Block



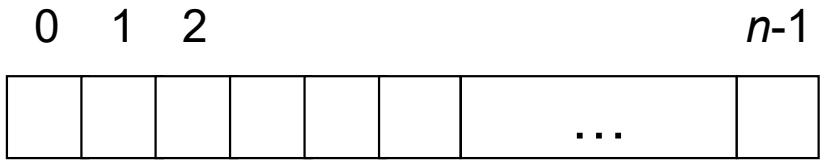
cylinder #, surface #,
track #, sector #



disk block #

Free-Space Management

Bit vector or bit map (*n* blocks)



$$\text{bit}[i] = \begin{cases} 1 \Rightarrow \text{block}[i] \text{ free} \\ 0 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

Bit map requires extra space. Example:

block size = 2^{12} bytes

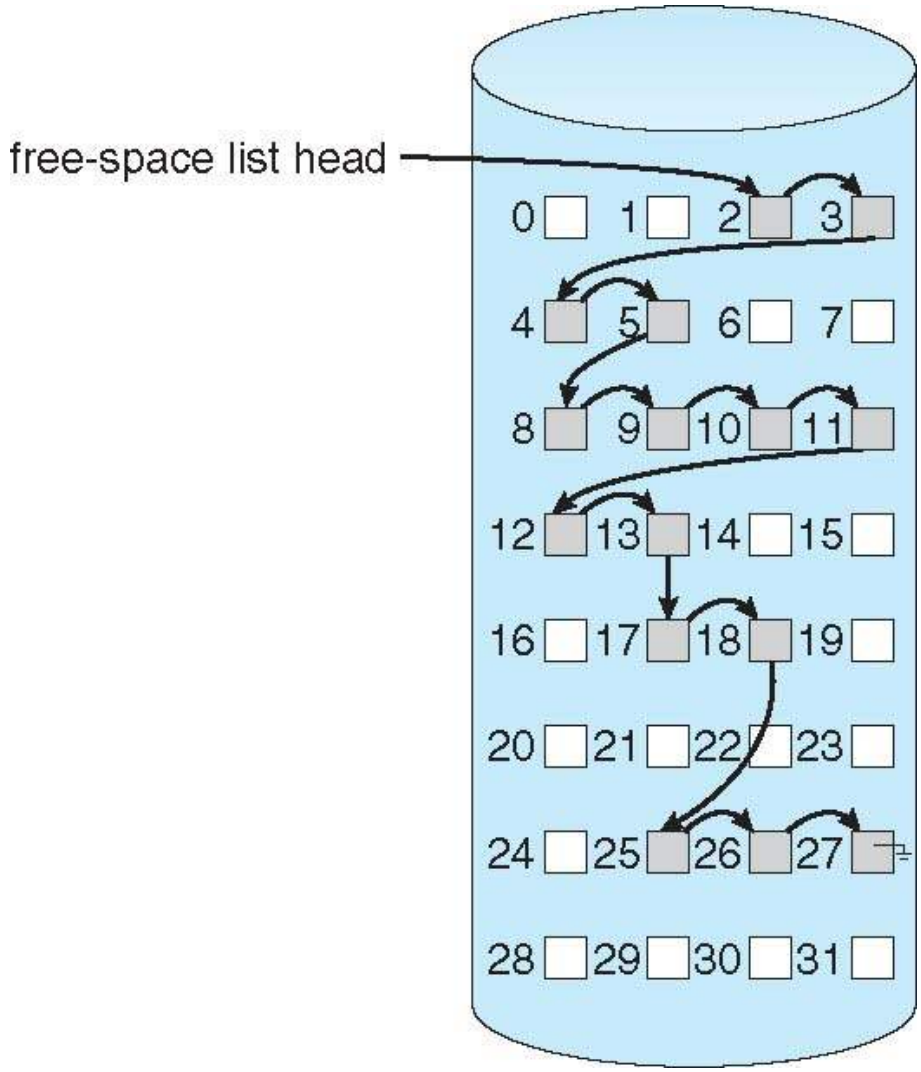
disk size = 2^{30} bytes (1 gigabyte)

$n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)

Easy to get contiguous files

Free-Space Management

Linked list (free list)



Grouping

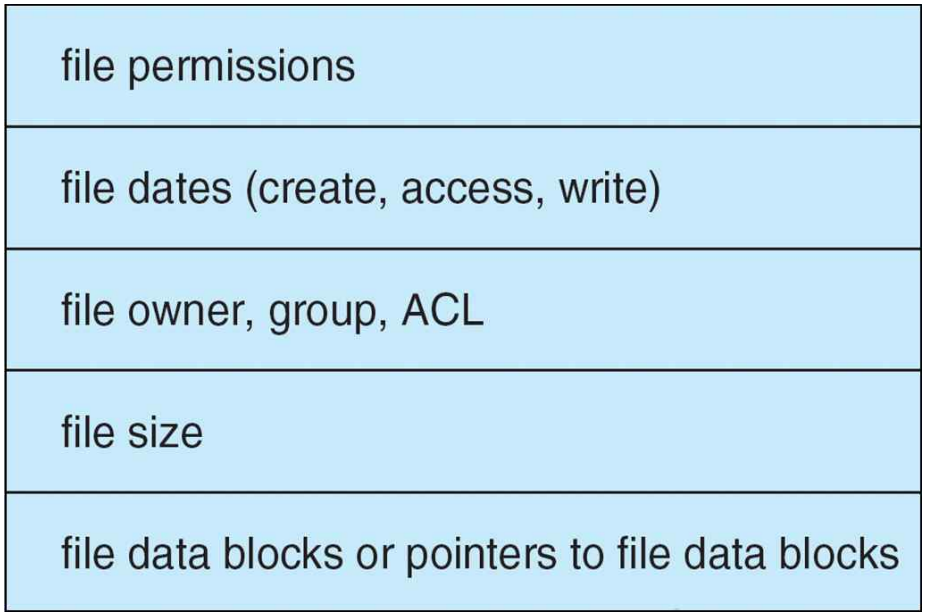
Counting

Space maps

- ✓ ZFS

A Typical File Control Block

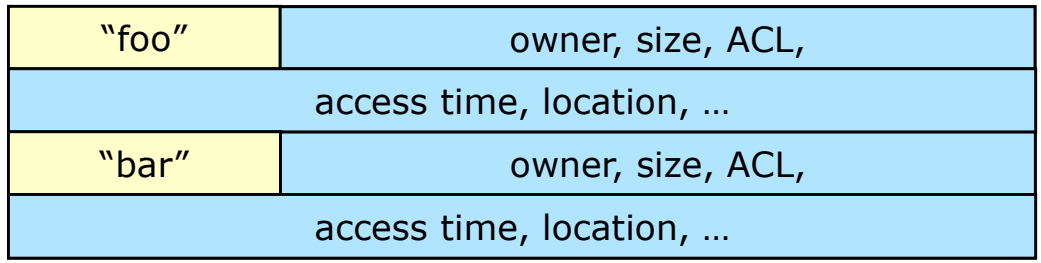
i-node (UFS) or in master file table (NTFS)



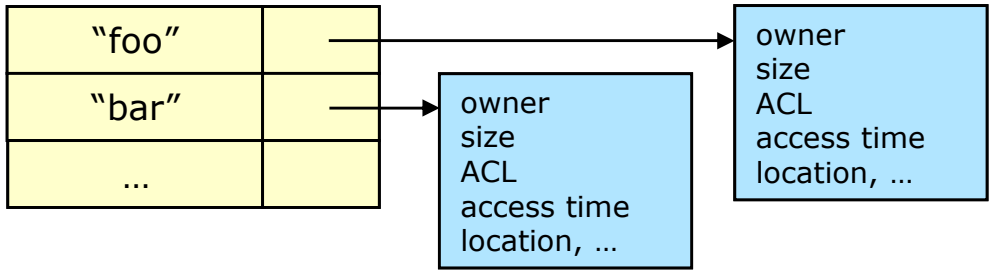
Directory Implementation

The location of metadata (FCB)

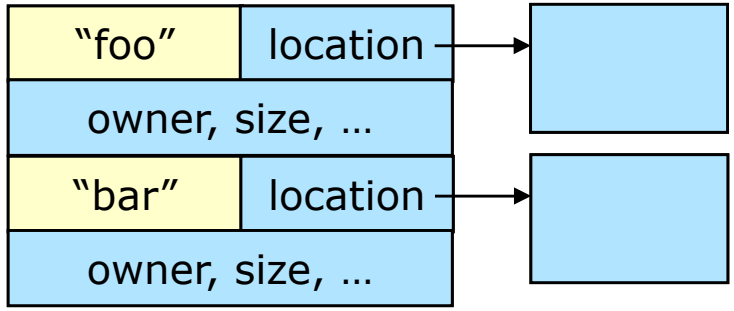
- ✓ In the directory entry



- ✓ In the separate data structure (e.g., i-node)



- ✓ A hybrid approach

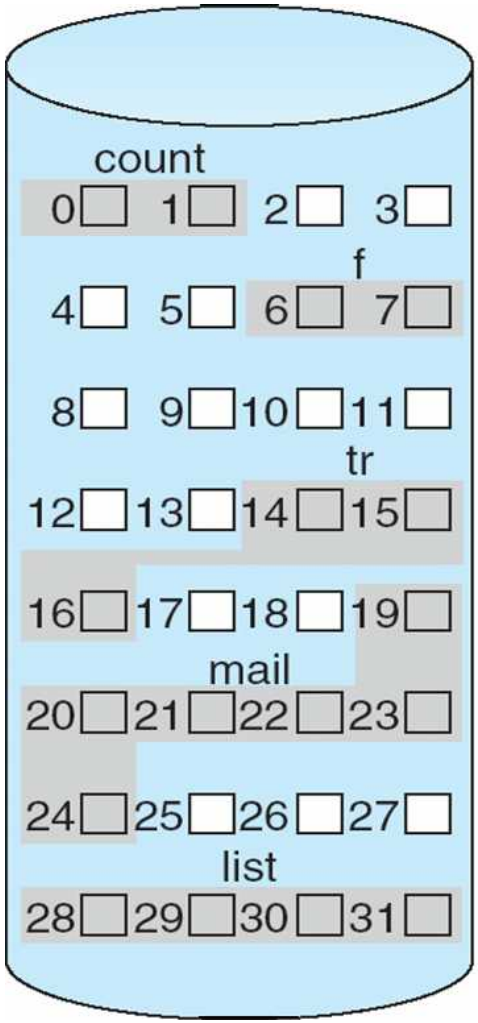


Allocation Methods

An allocation method refers to how disk blocks are allocated for files

- ✓ Contiguous allocation
- ✓ Linked allocation
- ✓ Indexed allocation

Contiguous Allocation



directory

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

Contiguous Allocation

Advantages

- ✓ The number of disk seeks is minimal
- ✓ Directory entries can be simple:
  <file name, starting disk block, length, etc.>

Disadvantages

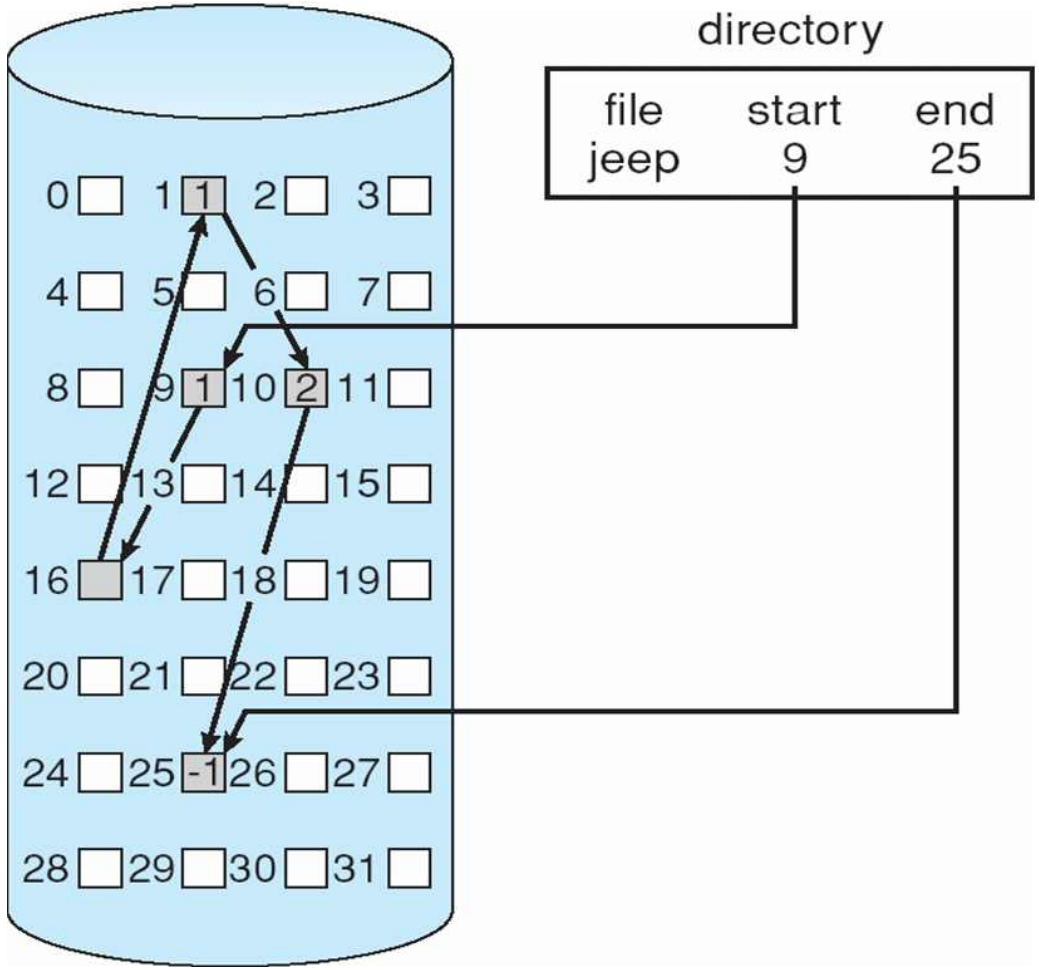
- ✓ Requires a dynamic storage allocation: First / best fit
- ✓ External fragmentation: may require a compaction
- ✓ The file size is hard to predict and varying over time

Feasible and widely used for CD-ROMS

- ✓ All the file sizes are known in advance
- ✓ Files will never change during subsequent use

Linked Allocation

block = pointer



Linked Allocation

Advantages

- ✓ Directory entries are simple:

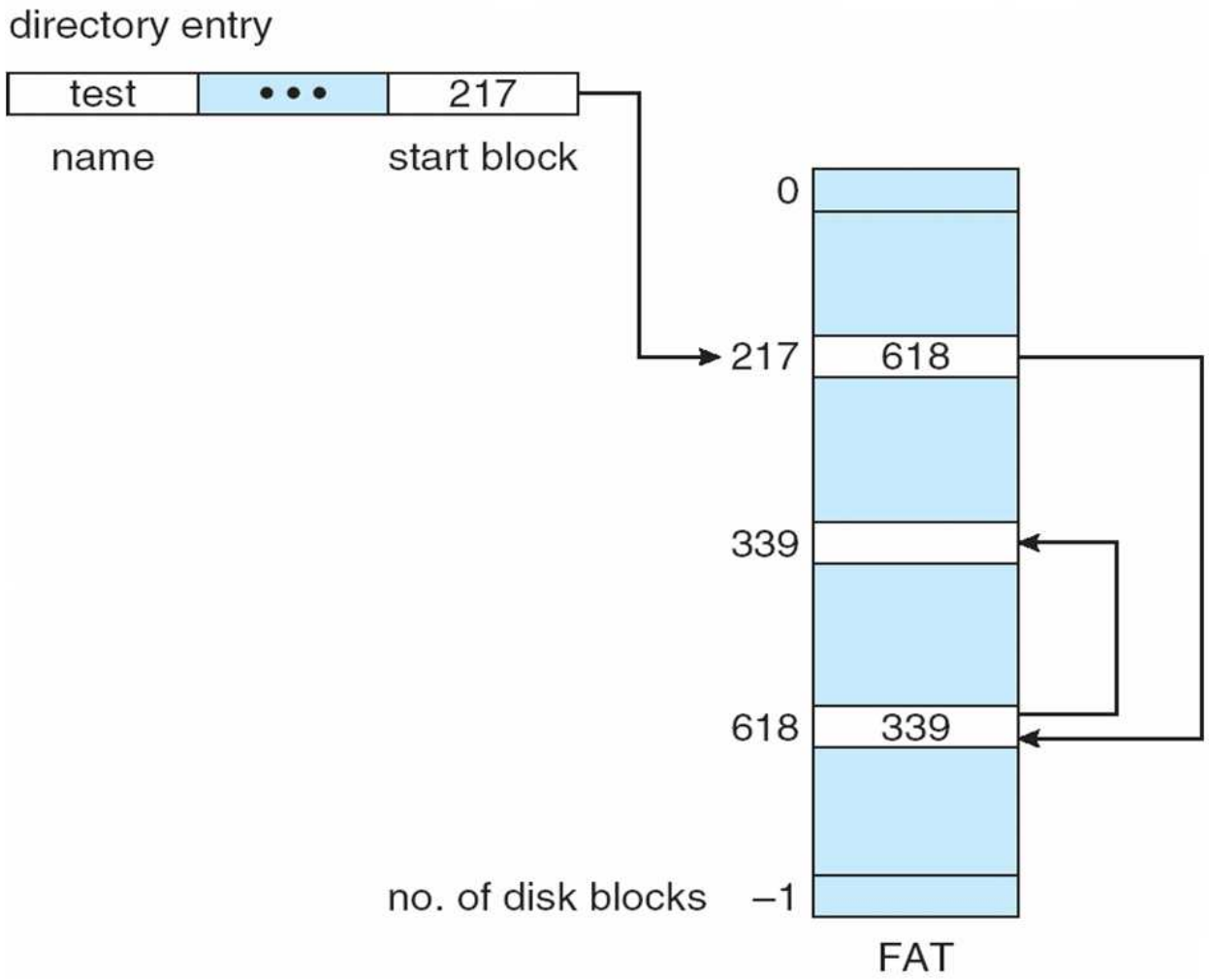
  <file name, starting block, ending block, etc.>
- ✓ No external fragmentation
  - the disk blocks may be scattered anywhere on the disk
- ✓ A file can continue to grow as long as free blocks are available

Disadvantages

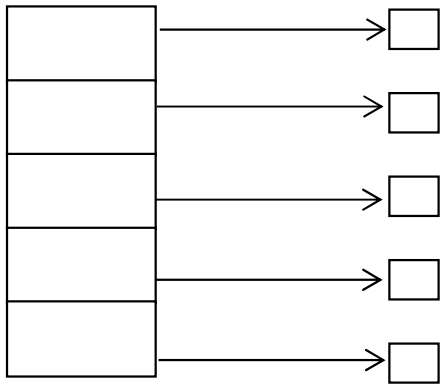
- ✓ It can be used only for sequentially accessed files
- ✓ Space overhead for maintaining pointers to the next disk block
- ✓ The amount of data storage in a block is no longer a power of two because the pointer takes up a few bytes
- ✓ Fragile: a pointer can be lost or damaged

Linked Allocation

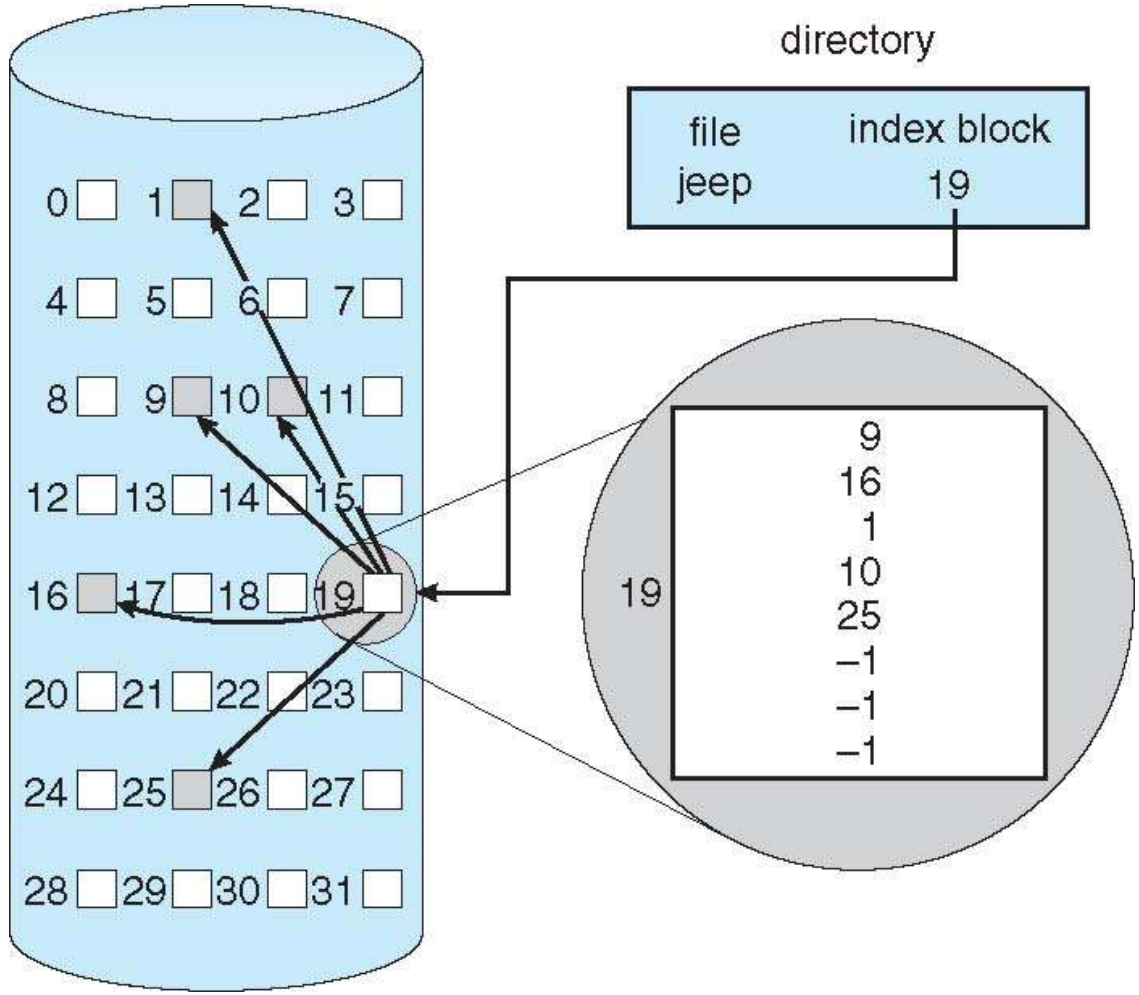
File-Allocation Table (FAT)



Indexed Allocation



index table



Indexed Allocation

Advantages

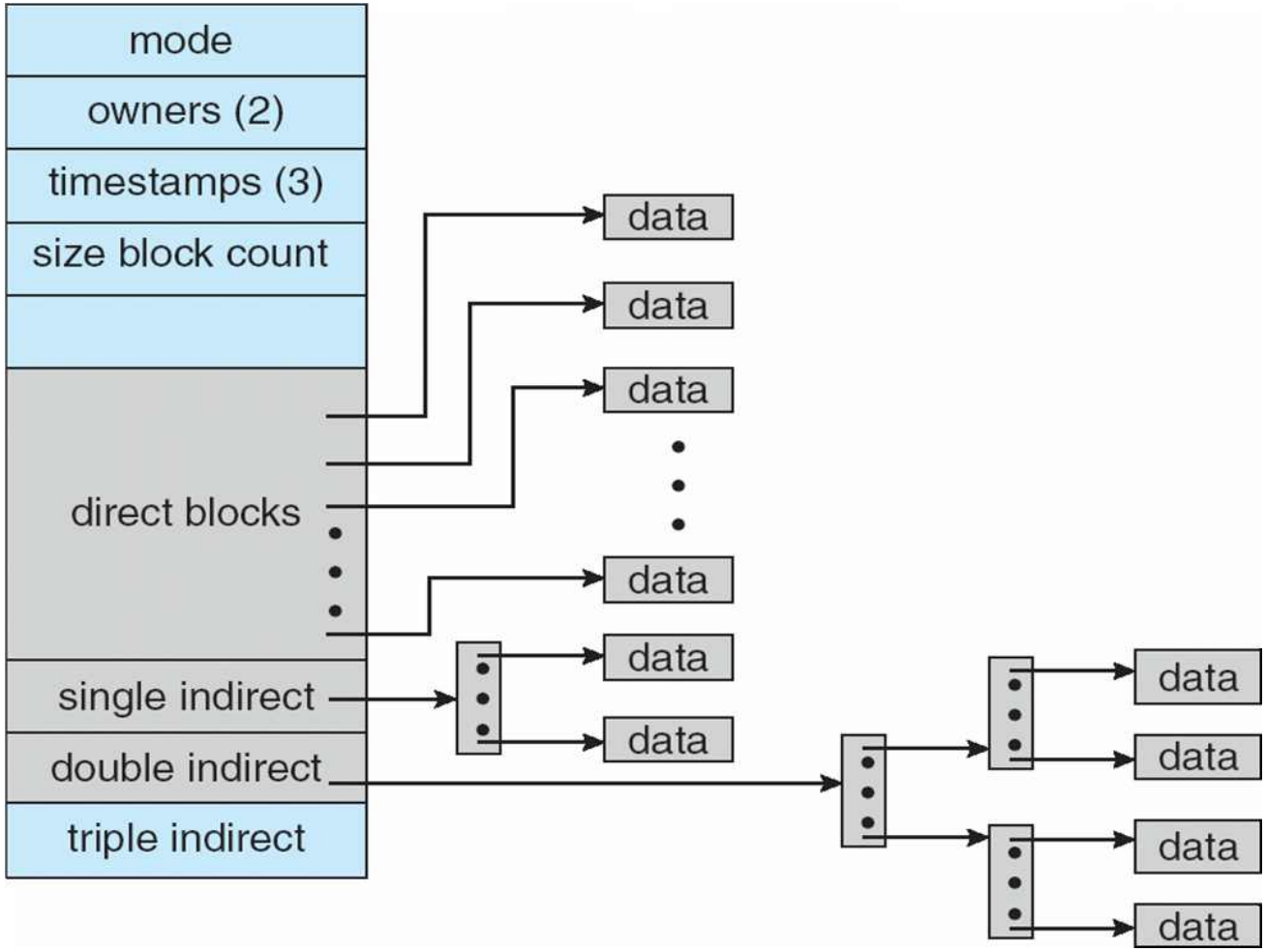
- ✓ Supports direct access, without suffering from external fragmentation
- ✓ I-node need only be in memory when the corresponding file is open

Disadvantages

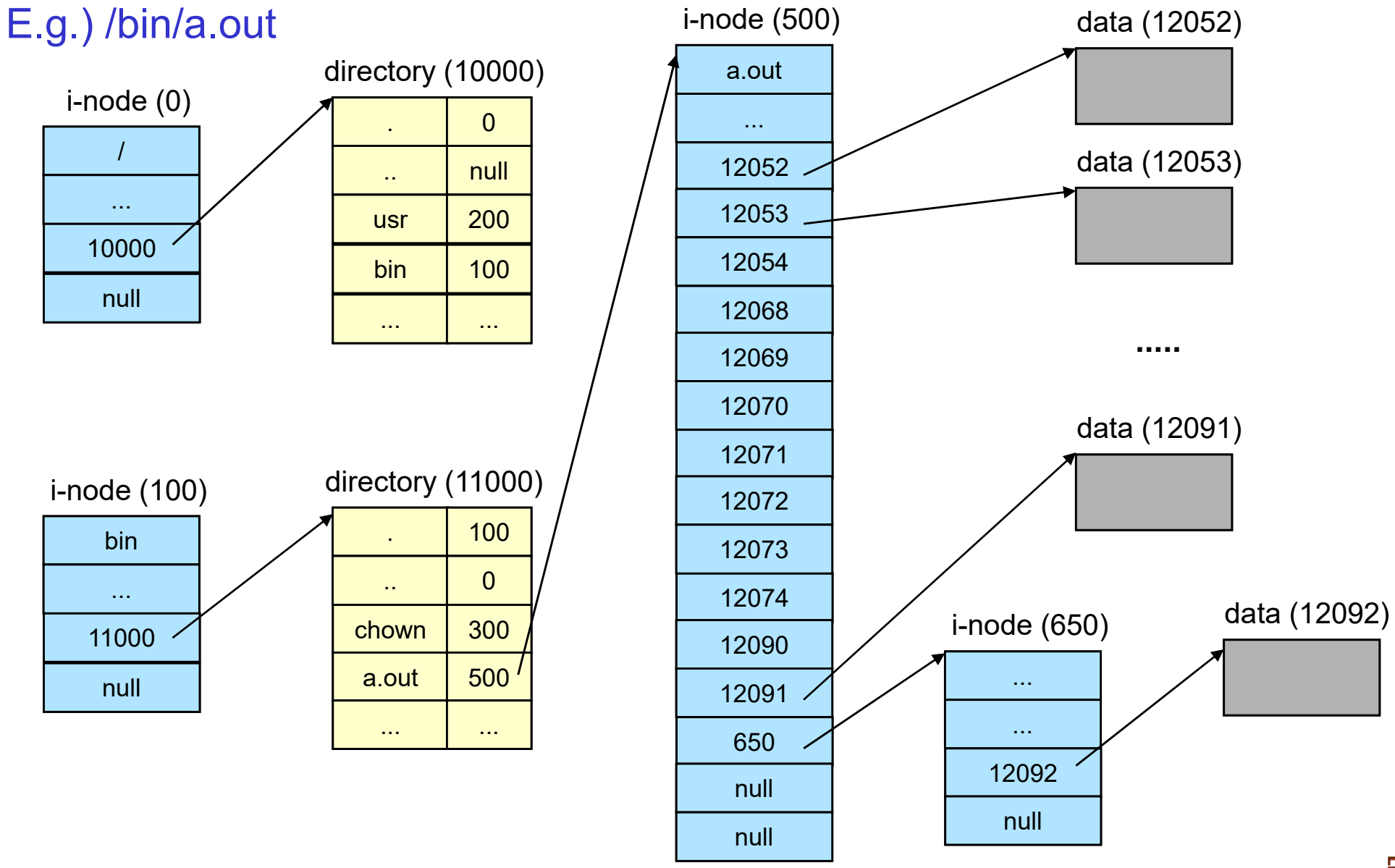
- ✓ Space overhead for indexes:
  - (1) Linked scheme: link several index blocks
  - (2) Multilevel index blocks
  - (3) Combined scheme: UNIX UFS
    - 12 direct blocks, single indirect block, double indirect block, triple indirect block

Indexed Allocation

Combined scheme: UFS (4K bytes per block, 32 bits addresses)



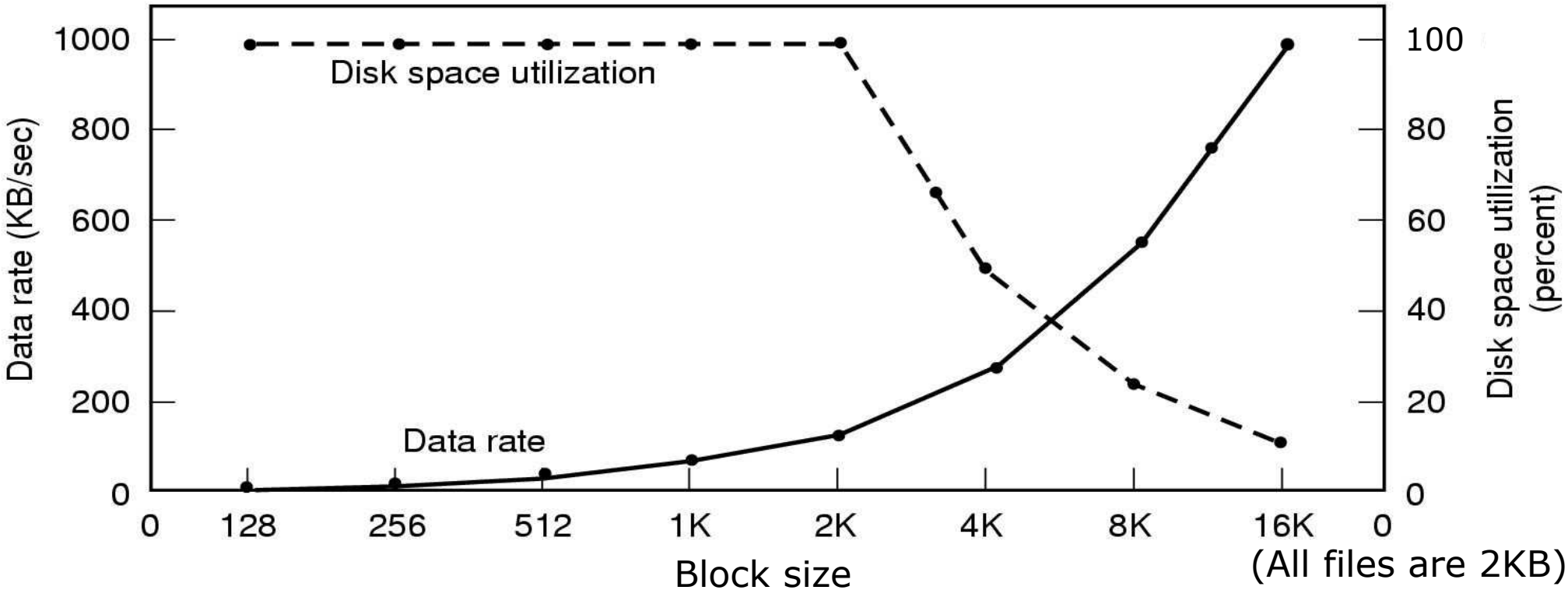
UNIX File System (UFS) Structure



Block Size Performance vs. Efficiency

Block size

- ✓ Disk block size vs. file system block size
- ✓ The median file size in UNIX is about 1KB



Read-Ahead

File system predicts that the process will request next block

- ✓ File system goes ahead and requests it from the disk
- ✓ This can happen while the process is computing on previous block, overlapping I/O with execution
- ✓ When the process requests block, it will be in cache

Compliments the disk cache, which also is doing read ahead

Very effective for sequentially accessed files

File systems try to prevent blocks from being scattered across the disk during allocation or by restructuring periodically

Cf) Free-behind

Buffer Cache

Applications exhibit significant locality for reading and writing files

Idea: cache file blocks in memory to capture locality in buffer cache (or disk cache)

- ✓ Cache is system wide, used and shared by all processes
- ✓ Reading from the cache makes a disk perform like memory
- ✓ Even a 4MB cache can be very effective

Issues

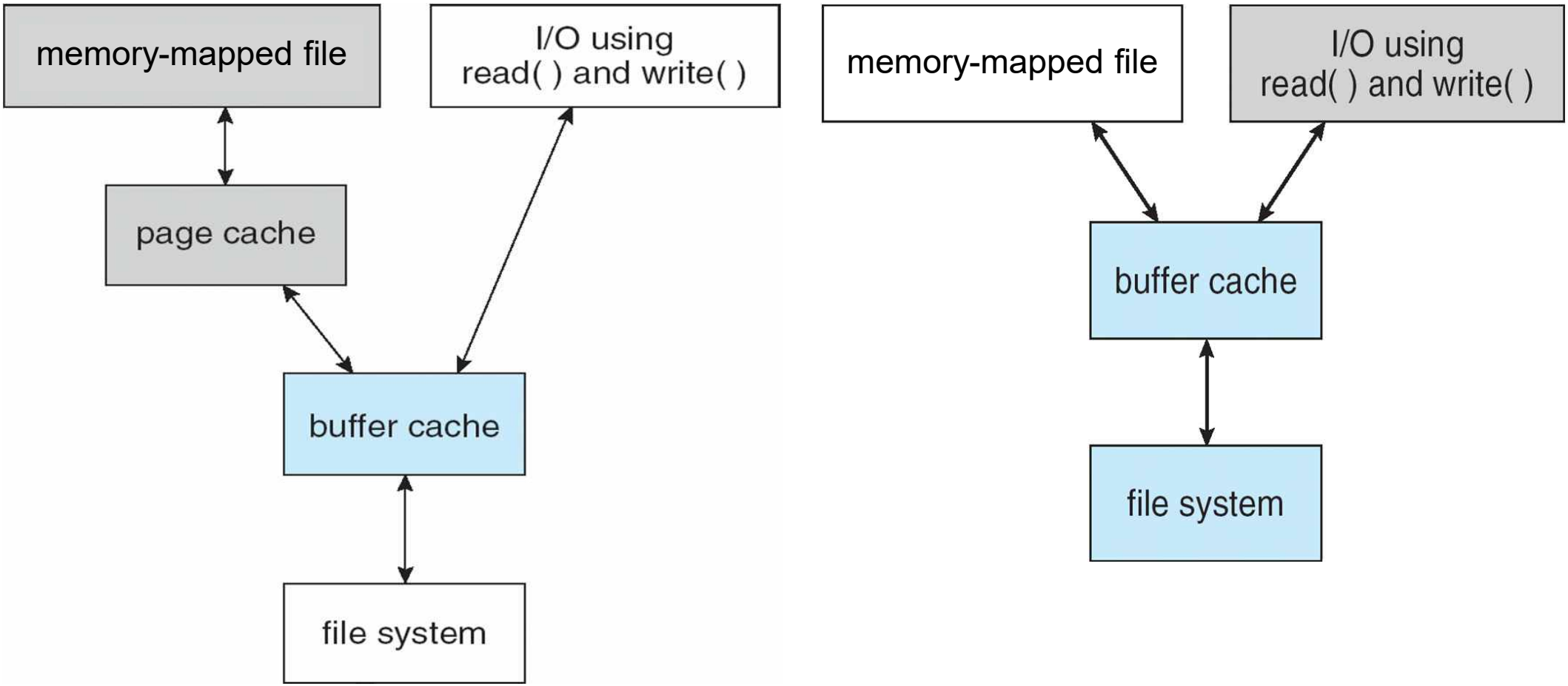
- ✓ The buffer cache competes with VM
- ✓ Like VM, it has limited size
- ✓ Need replacement algorithms again

  (References are relatively infrequent, so it is feasible to keep all the blocks in exact LRU order)

Unified Buffer Cache

Page cache for demand paging (including memory-mapped file)

Buffer cache for ordinary file system I/O



Caching Writes

Synchronous writes are very slow

Asynchronous writes (or write-behind, write-back)

- ✓ Maintain a queue of uncommitted blocks
- ✓ Periodically flush the queue to disk
- ✓ Unreliable: metadata requires synchronous writes (with small files, most writes are to metadata)

Reliability

File system consistency

- ✓ File system can be left in an inconsistent state if cached blocks are not written out due to the system crash
- ✓ It is especially critical if some of those blocks are i-node blocks, directory blocks, or blocks containing the free list
- ✓ Most systems have a utility program that checks file system consistency
  - Windows: scandisk
  - UNIX: fsck

Log Structured File Systems

Journaling file systems

- ✓ Fsck'ing takes a long time, which makes the file system restart slow in the event of system crash
- ✓ Record a log, or journal, of changes made to files and directories to a separate location (preferably a separate disk)
- ✓ If a crash occurs, the journal can be used to undo any partially completed tasks that would leave the file system in an inconsistent state
- ✓ IBM JFS for AIX, Linux

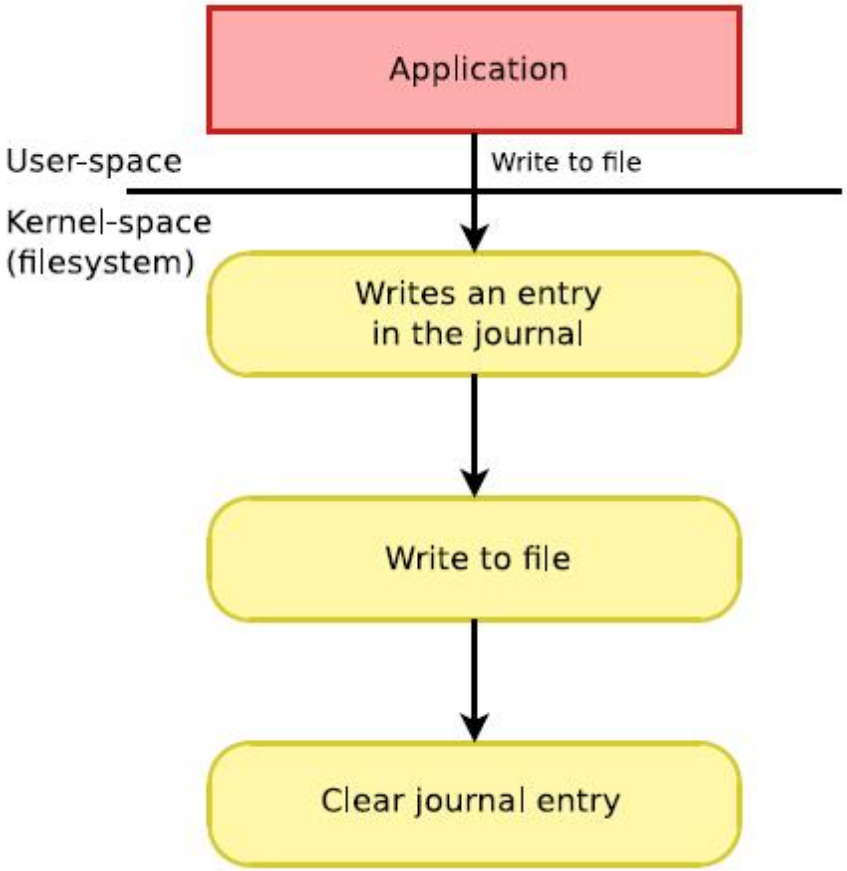
  Veritas VxFS for Solaris, HP-UX, Unixware, etc.

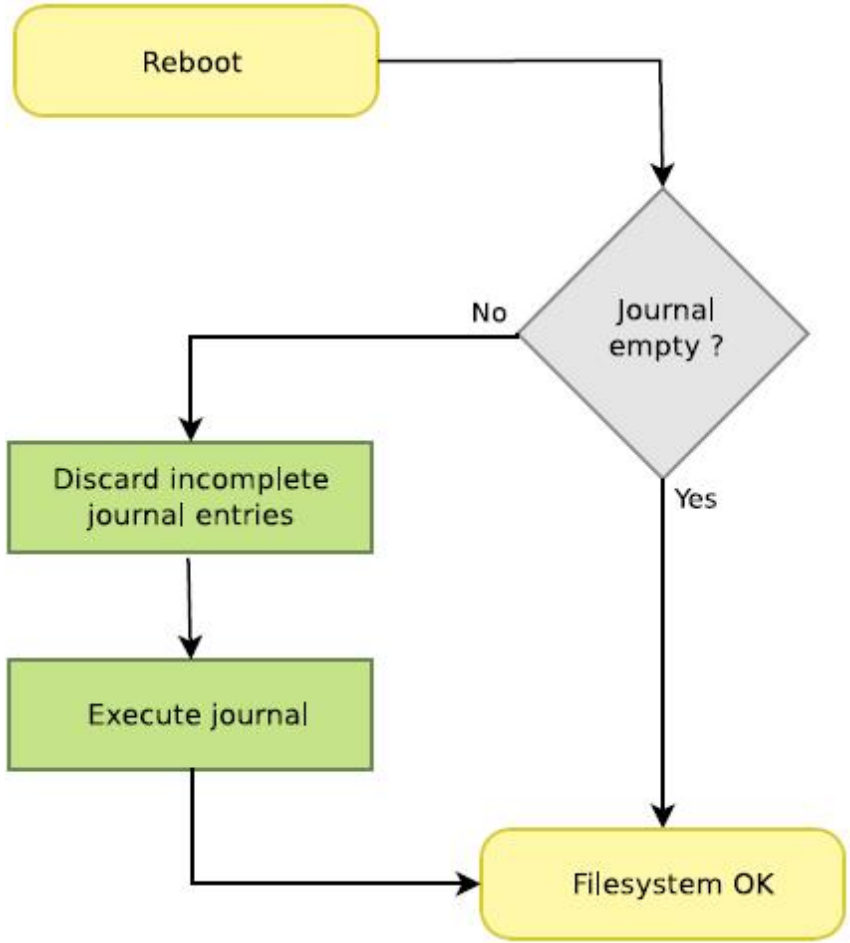
  SGI XFS for IRIX, Linux

  Reiserfs, ext3 for Linux

  NTFS for Windows

Journaling File Systems





Remote File Systems

Network File Systems (NFS)