File Management

Chapter 12

File System Properties

- Long-term existence
- Sharable between processes
- Structure

File Operations

- Create
- Delete
- Open
- Close
- Read
- Write

Terms Used with Files

- Field
  - Basic element of data
  - Contains a single value (e.g. student last name, date of birth..)
  - Characterized by its length and data type
  - Fixed or variable length
- Record
  - Collection of related fields
  - Fixed or variable length (if contains fields of variable length)
  - Treated as a unit
    - Example: student record

Terms Used with Files

- File
  - Collection of similar records
  - Treated as a single entity
  - Have file names
  - Access control apply at file level
- Database
  - Collection of related data
  - Relationships exist among elements
  - Consists of one or more file types

Typical Operations

- Retrieve_All– records of file
- Retrieve_One—record
- Retrieve_Next—logically after the most recently retreived record
- Retrieve_Previous
- Insert_One
- Delete_One
- Update_One
- Retrieve_Few—that sattisfy a certain criteria

File Management Systems

- The way a user or application may access files
- Programmer does not need to develop file management software

Objectives for a File Management System

- Meet the data management needs and requirements of the user
- Guarantee that the data in the file are valid
- Optimize performance
- Provide I/O support for a variety of storage device types

Objectives for a File Management System

- Minimize or eliminate the potential for lost or destroyed data
- Provide a standardized set of I/O interface routines
- Provide I/O support for multiple users

Minimal Set of Requirements

- Each user should be able to **create, delete, read, write and modify** files
- Each user may have **controlled access to files of others**
- Each user may **control** what type of **accesses** are allowed to his own files
- Each user should be able to **restructure** the user's files in a form appropriate to the problem

Minimal Set of Requirements

- Each user should be able to **move data between files**
- Each user should be able to **back up and recover** the user's files in case of damage
- Each user should be able to **access** the user's files by using **symbolic names**

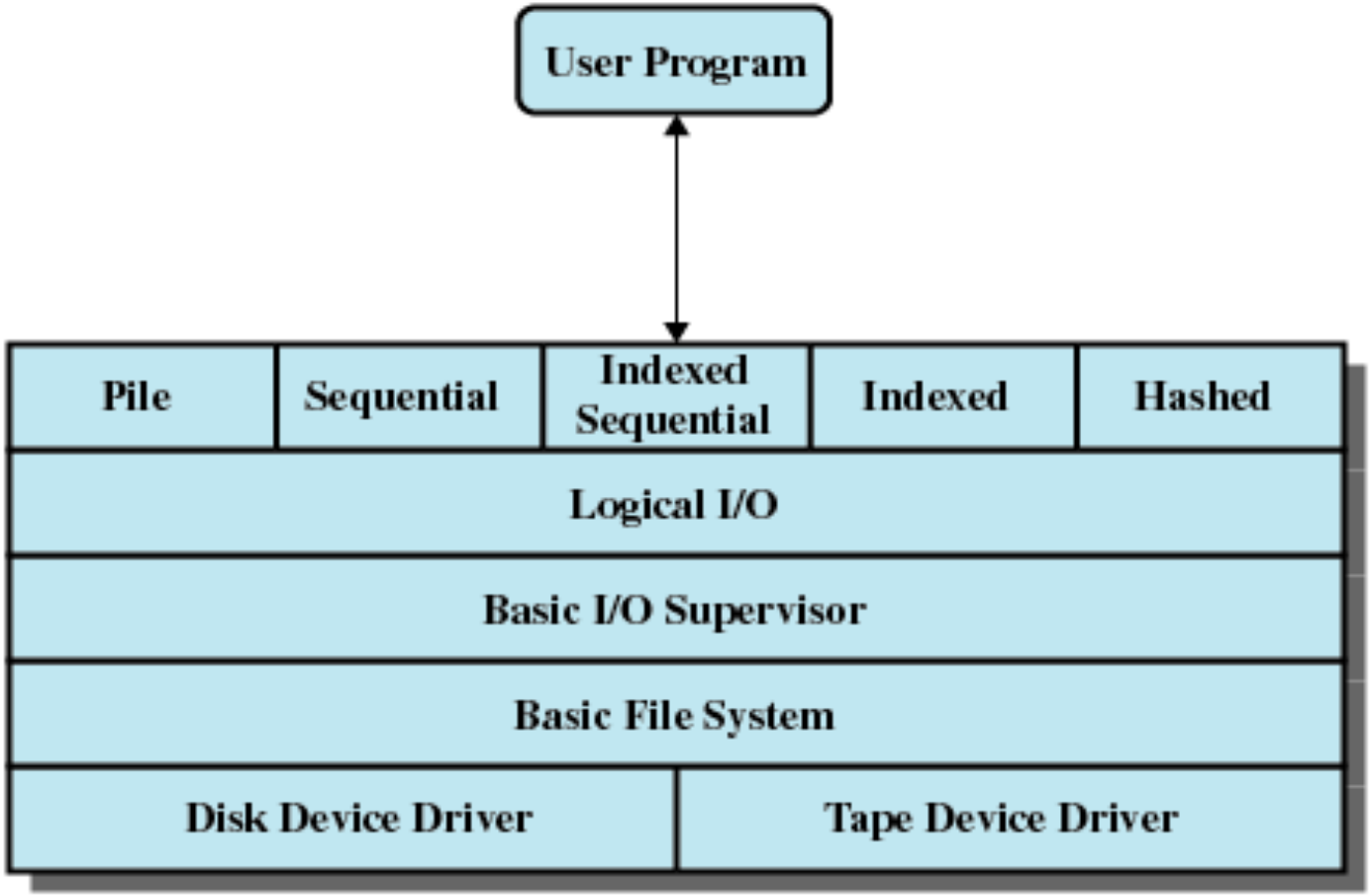


Figure 12.1 File System Software Architecture

Device Drivers

- Lowest level
- Communicates directly with peripheral devices
- Responsible for starting I/O operations on a device
- Processes the completion of an I/O request

Basic File System

- Physical I/O
- Deals with exchanging blocks of data
- Does not understand the content or structure of data
- Concerned with the placement of blocks
- Concerned with buffering blocks in main memory

Basic I/O Supervisor

- Responsible for file I/O initiation and termination
- Control structures are maintained
- Concerned with selection of the device on which file I/O is to be performed
- Concerned with scheduling access to optimize performance
- Part of the operating system

Logical I/O

- Enables users and applications to access records
- Provides general-purpose record I/O capability
- Maintains basic data about file

Access Method

- Provides standard interfaces btw applications and filesystems & devices
- Reflect different file structures
- Different ways to access and process data

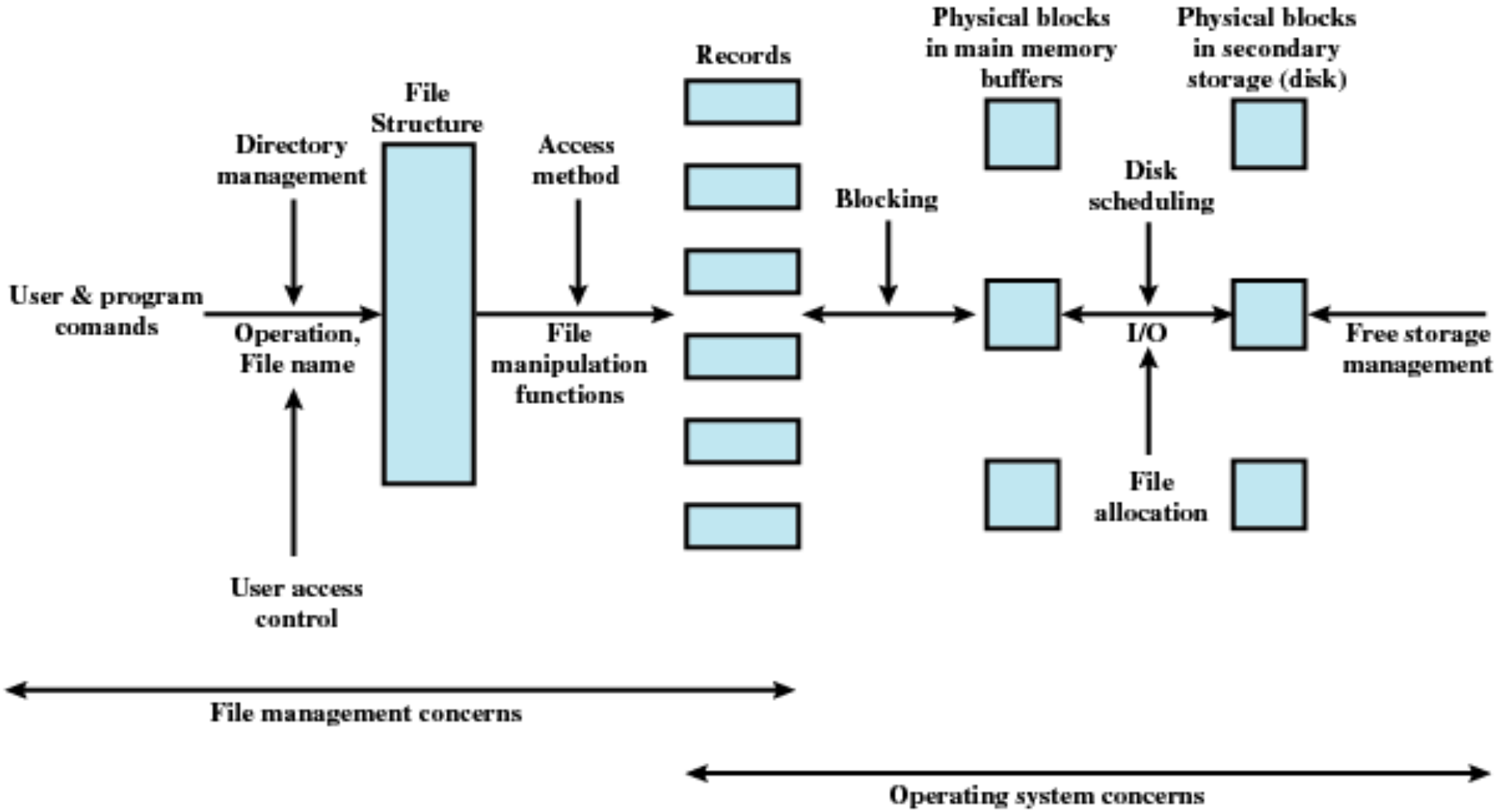


Figure 12.2 Elements of File Management

File Directories

- Contains information about files
  - Attributes
  - Location
  - Ownership
- Directory itself is a file owned by the operating system
- Provides mapping between file names and the files themselves

Elements of a File Directory

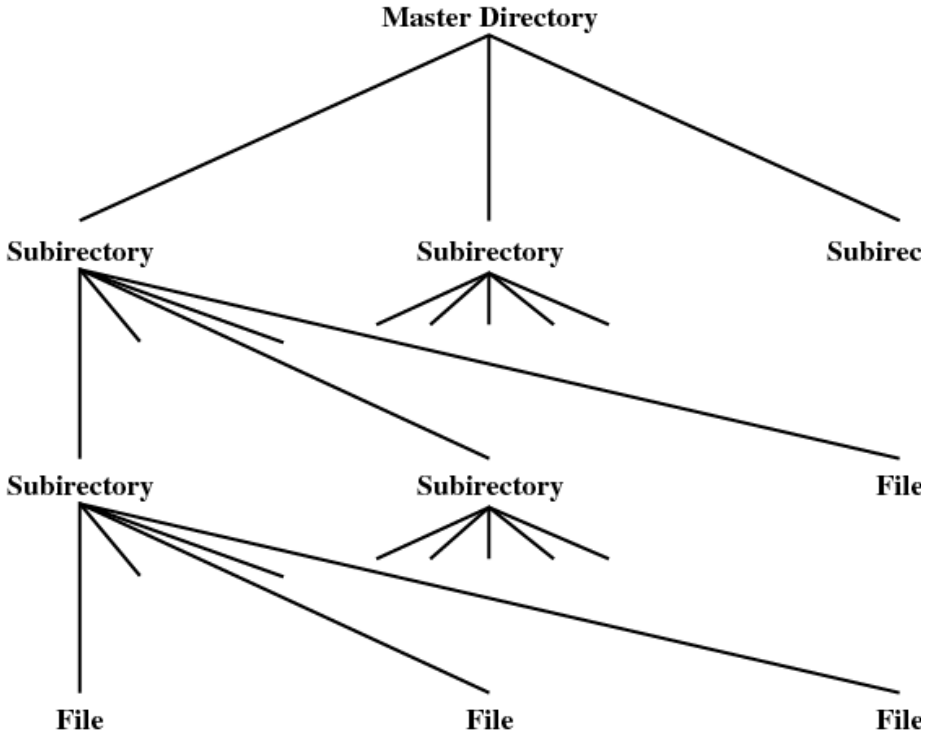
- Basic Info
  - File Name, type, organization
- Address Information
  - Volume (storage device)
  - Starting Address—physical address on secondary storage (e.g. track & block number)
  - Size used (words/bytes/blocks) and allocated (max size)
- Access Control Information
  - Owner, Access Info (user name &passwd for each user), Permitted actions (R/W/X, NW transmission)
- Usage Information
  - Date Created/Last Read Access/Last Modified/Last Backup with corresponding user identities
  - Current usage (processes that are accessing it and how, possible locks, updates in MM but not in disk)

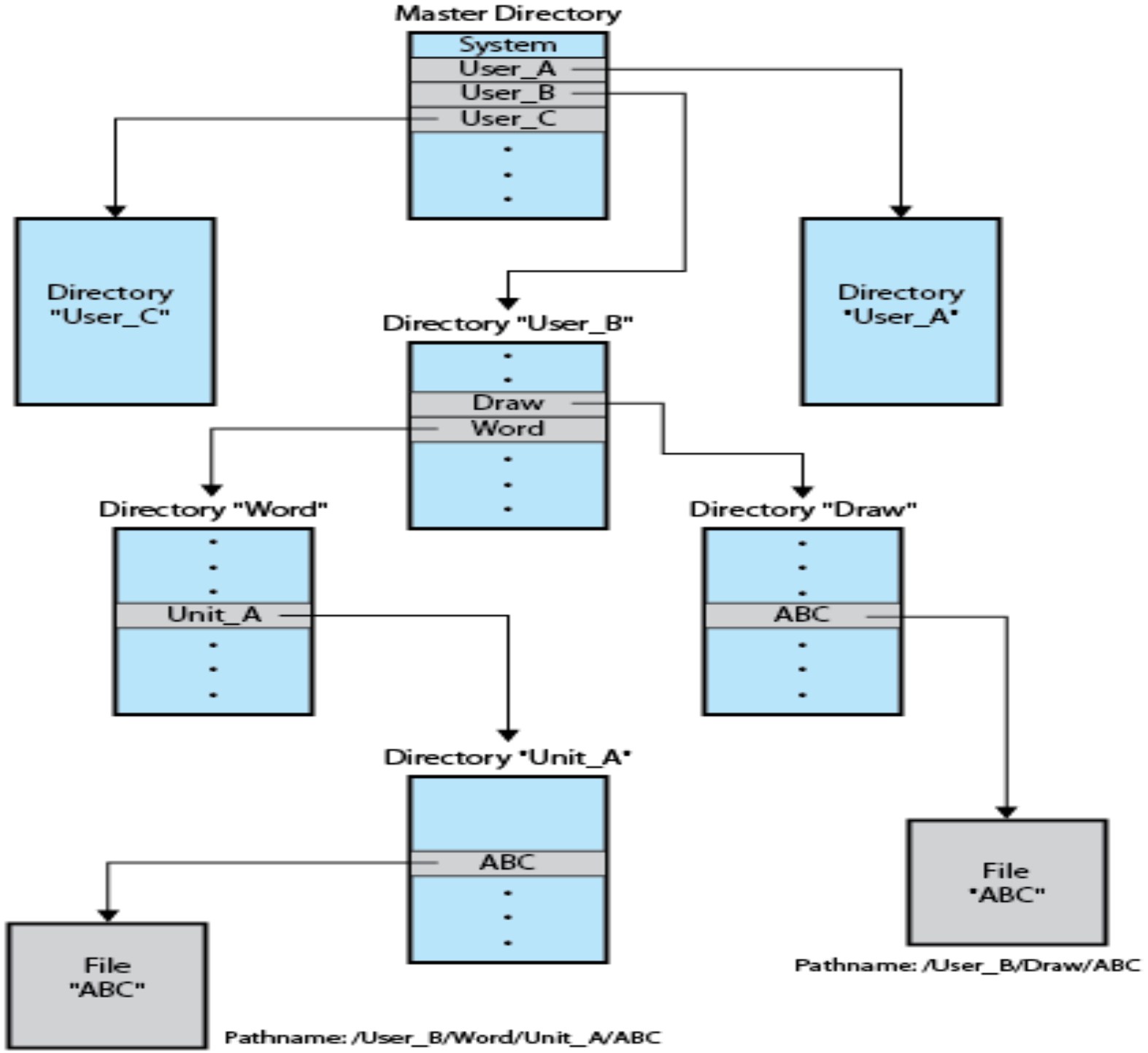
Two-level Scheme for a Directory

- One directory for each user and a master directory
- Master directory contains entry for each user
  - Provides address and access control information
- Each user directory is a simple list of files for that user
- Problems with structuring (list by type, date created/ modified and so on/ listing only some filetypes e.g. .pdfs/ all names need to be unique)

Tree-Structured Directory

- Master directory with user directories underneath
- Each user directory may have subdirectories and files as entries
- Files->paths
- Same filename allowed;
- Current dir is working dir
- Files referenced rel. to working dir




Master Directory
System
User_A
User_B
User_C
Directory "User_C"
Directory "User_B"
Directory "User_A"
Draw
Word
Directory "Word"
Directory "Draw"
Unit_A
ABC
Directory "Unit_A"
ABC
File "ABC"
File "ABC"
Pathname: /User_B/Word/Unit_A/ABC
Pathname: /User_B/Draw/ABC

Secondary Storage Management

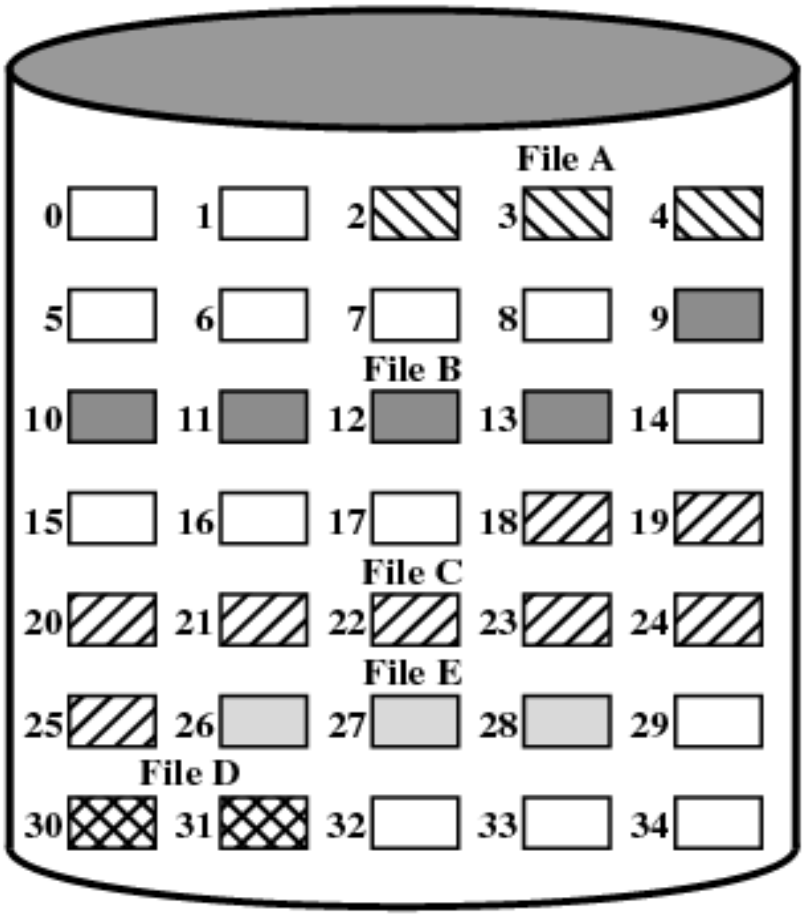
- Space must be allocated to files
- Must keep track of the space available for allocation

Preallocation vs Dynamic allocation

- Preallocation:
  - Need the maximum size for the file at the time of creation
  - Difficult to reliably estimate the maximum potential size of the file
  - Tend to overestimated file size so as not to run out of space
- Dynamic allocation: File portions
  - block division vs entire file
  - Fixed portions vs dynamic portions

Methods of File Allocation

- Contiguous allocation
  - Single set of blocks is allocated to a file at the time of creation
  - Only a single entry in the file allocation table
    - Starting block and length of the file
- External fragmentation will occur
  - Need to perform compaction



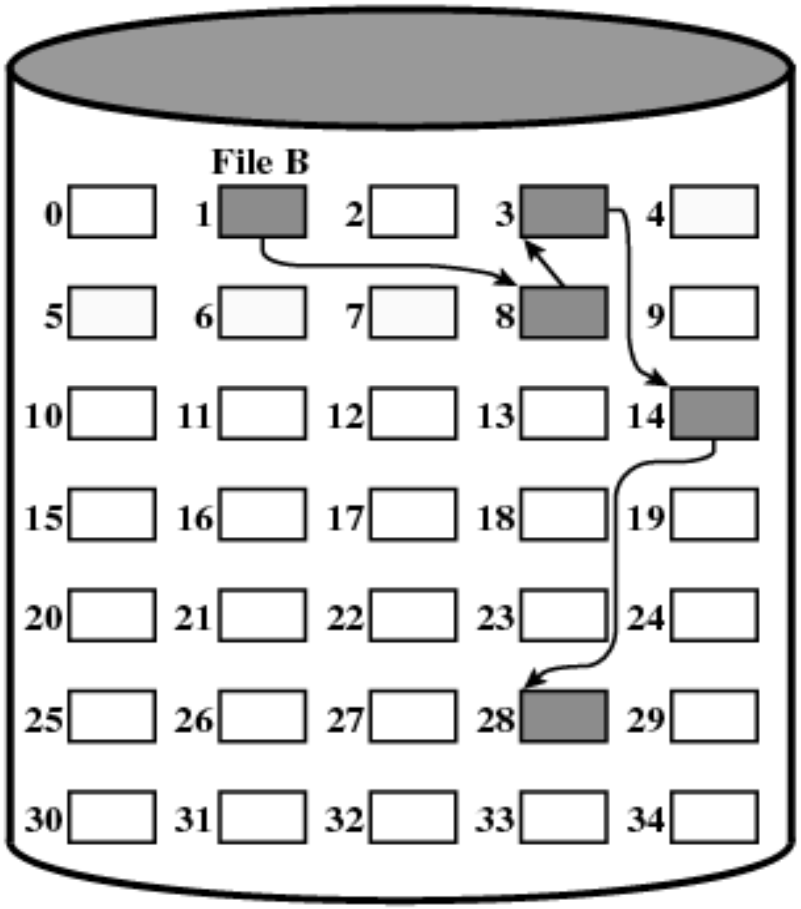
File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Figure 12.7 Contiguous File Allocation

Methods of File Allocation

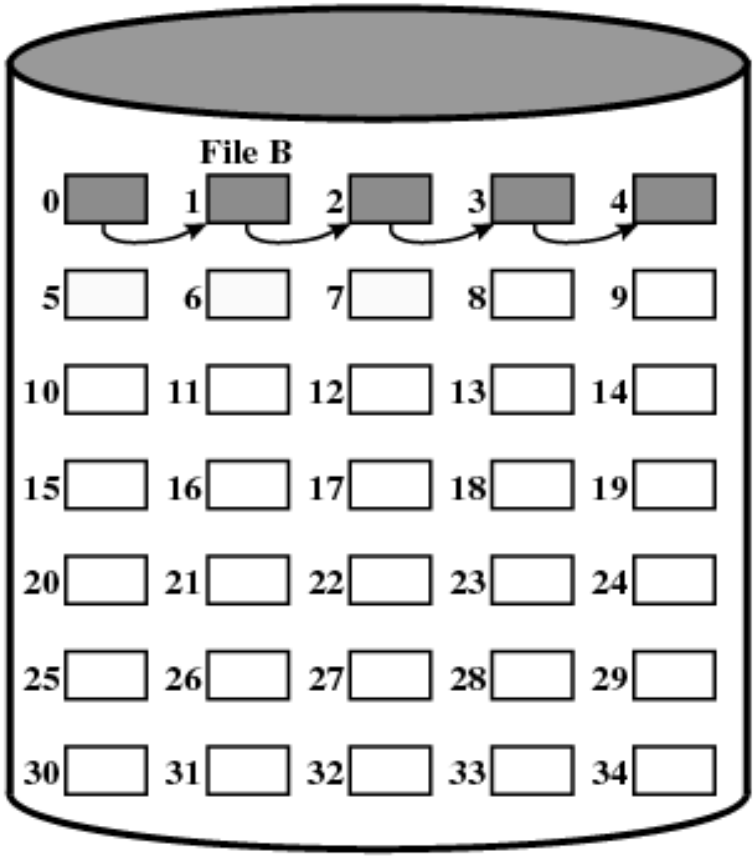
- Chained allocation
  - Allocation on basis of individual block
  - Each block contains a pointer to the next block in the chain
  - Only single entry in the file allocation table
    - Starting block and length of file
- No external fragmentation
- No accommodation of the principle of locality (several disk accesses to bring in several blocks block)
- Some systems perform Consolidation to overcome the problem



File Allocation Table

File Name	Start Block	Length
...	...	...
File B	1	5
...	...	...

Figure 12.9 Chained Allocation



File Allocation Table

File Name	Start Block	Length
...	...	...
File B	0	5
...	...	...

Figure 12.10 Chained Allocation (After Consolidation)

Methods of File Allocation

- Indexed allocation
  - File allocation table contains a separate one-level index for each file
  - The index has one entry for each portion allocated to the file
  - The file allocation table contains block number for the index

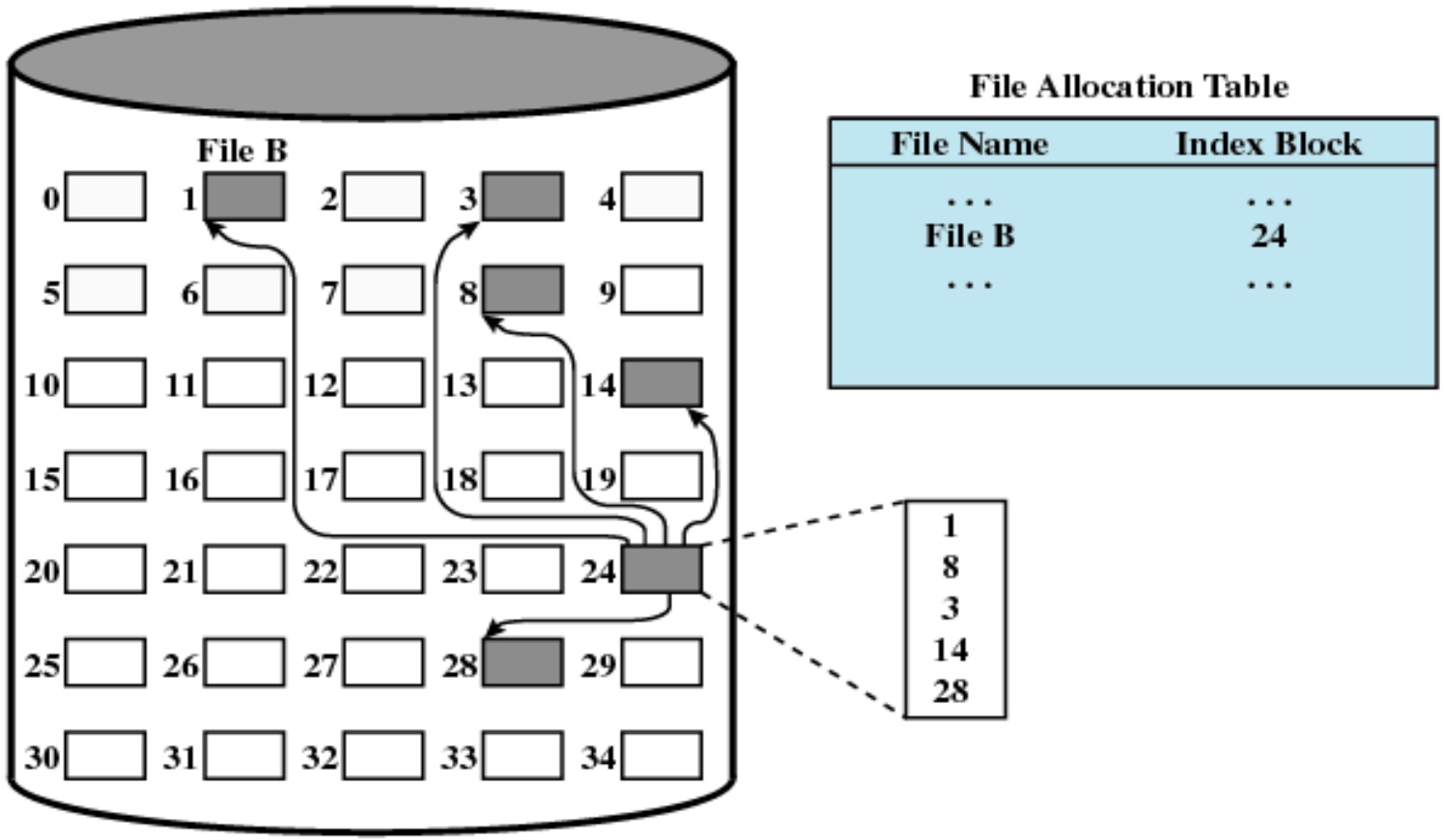
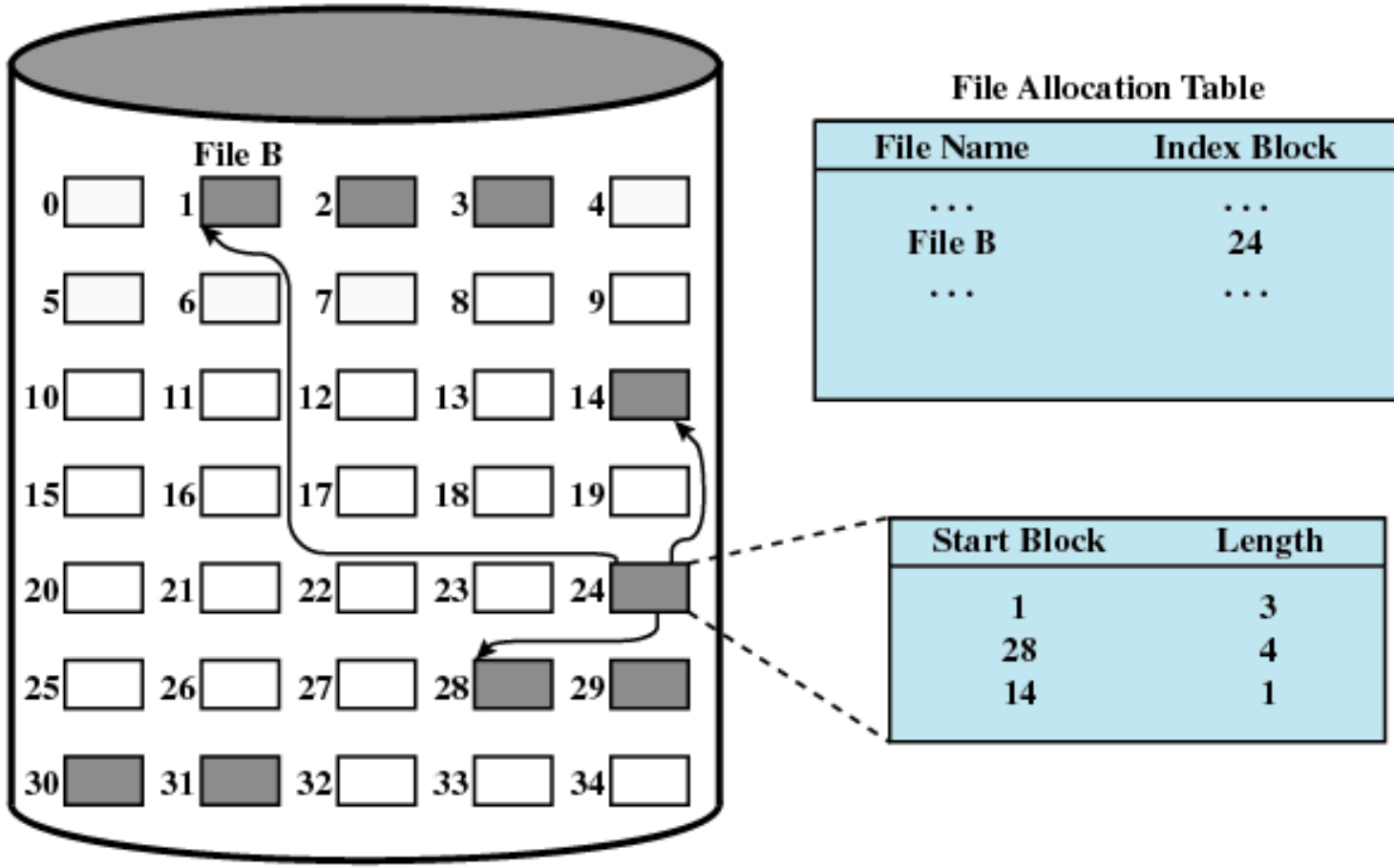


Figure 12.11 Indexed Allocation with Block Portions



gure 12.12 Indexed Allocation with Variable-Length Portio

Free Space Management

- Bit Tables
  - 1 bit for each block: 0 free; 1 allocated.
  - Can be considerably big to be kept in main memory
  - Will then need many access to disk for search
- Chained Free Portions (pointer + length value)
  - Prone to fragmentation
- Indexing—one entry for each free portion
- Free Block List
  - Cannot be kept in MM (considerably larger than bitmap)
  - Space devoted is less than 1% of disk
  - Can be partially stored on MM (push-down stack or FIFO)

UNIX File Management

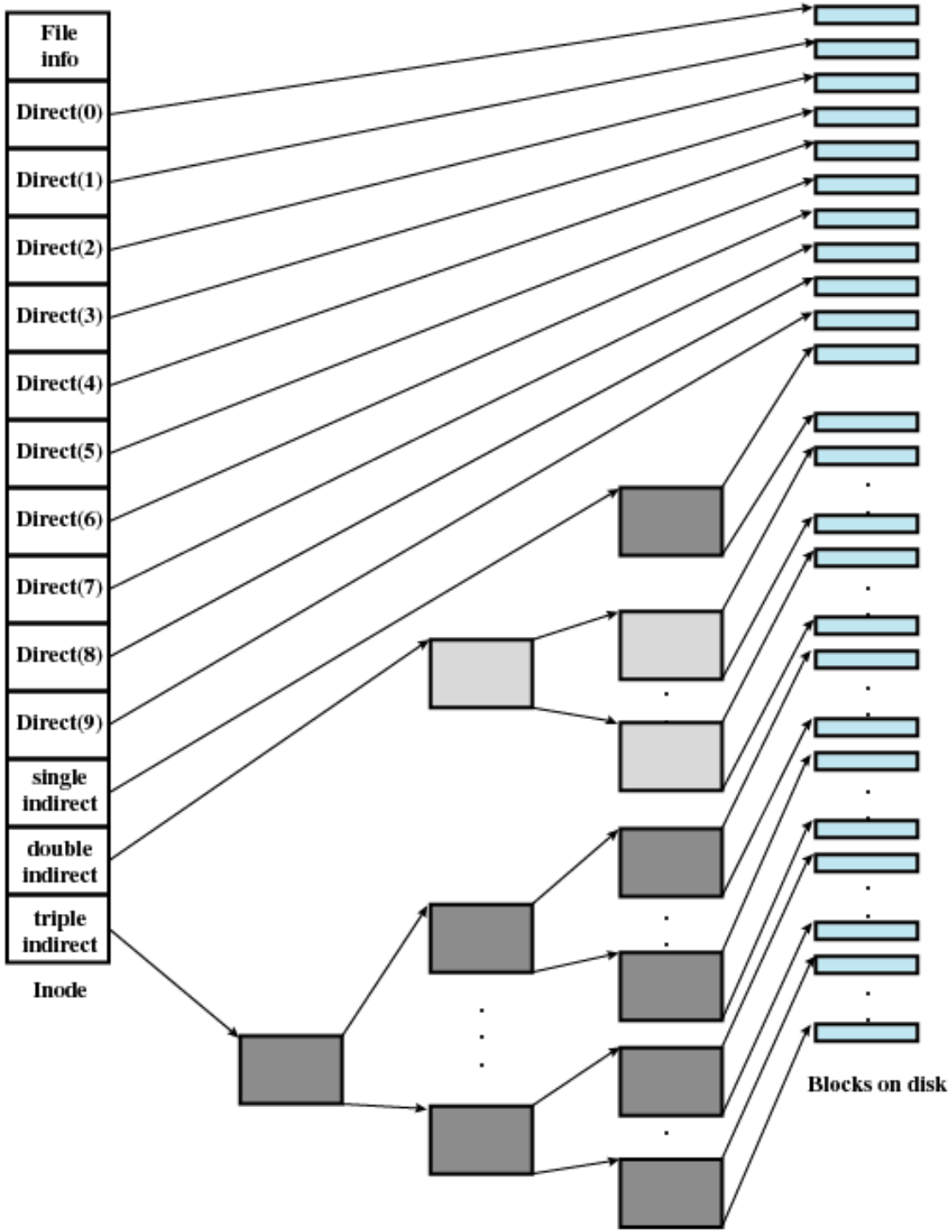
- Types of files
  - Regular, or ordinary—data as streams of bytes
  - Directory—list of filenames + pointers to inodes; editable by OS; readable by user programs
  - Special—mechanism to map phy dev to files (e.g. I/O devs)
  - Named pipes—IPC facility; buffers data; readable in FIFO fashion
  - Links—alterative filename for a file
  - Symbolic links—a datafile containing the name of the file linked to

Index nodes (inodes)

- Control structure that contains key information for a particular file
- One inode per file; but:
- Several filenames assoc to a unique inode.
- Inode table or list resides on disk.
- When file accessed its iNode its brought in MM

Inode structure

- Type&access mode
- File’s owner&group-access ids
- Times: creation;last write&read;last inode update by the OS
- Size of file in bytes
- Sequence of block pointers
- Nr of phy blocks used by the file (both data and attributes)
- Nr. of references to the file
- Flags describing file characteristics (kernel&user settable)
- Block size of data-blocks referenced by inode (same as the filesystem blocksize)
- Size of extended attribute information
- Extended attribute entries (e.g. access control lists or security labels for mandatory access control schemes)


File info
Direct(0)
Direct(1)
Direct(2)
Direct(3)
Direct(4)
Direct(5)
Direct(6)
Direct(7)
Direct(8)
Direct(9)
single indirect
double indirect
triple indirect
Inode
Blocks on disk

File allocation & Advantages

- Done on block-bases
- Is dynamic—not all blocks are contiguous
- Indexed method to keep track of pieces of file;
- Inodes: some direct pointers, 3 indirect pointers (single, double, triple)
- FreeBSD (4KB block size)—120 bytes of address information:
  - 12 direct pointers to first 12 blocks of the file (12 blocks, 48K size)
  - 13th pointer -> single indirect block (512 block addresses, 2M size)
  - 14th pointer -> double indirect block (256K block addresses, 1G size)
  - 15th pointer -> triple indirect block (128M block addresses, 521G size)
- **Advantages:**
  - Inode is of fixed size & small—easily kept in MM
  - Reduced access time (little or no indirection) 4 small files
  - Max file size large enough to satisfy all apps

Directories:

- Structured in hierarchical tree
- Contain files and/or other directories (subdirectories)
- Simple files containing lists of filenames and pointers to corresp. Inodes (the i-number)
- The i-number indexes the inode table

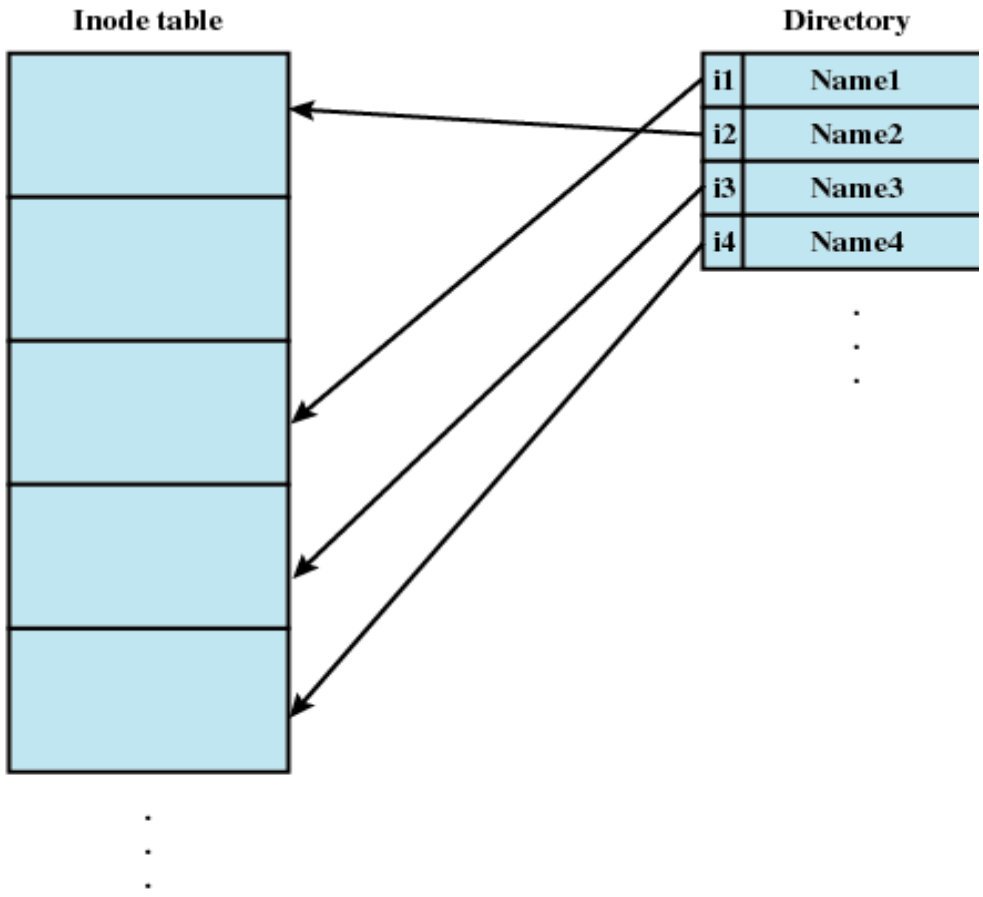
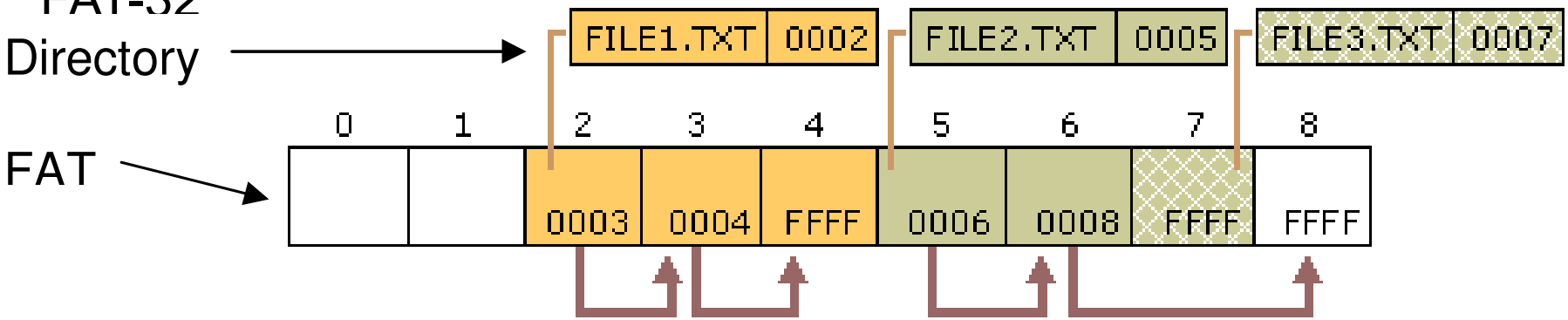


Figure 12.14 UNIX Directories and Inodes

FAT: File Allocation Table

- Linear table with 1 column; element-size is of 12/16/32 bits
- Usage and allocation status of blocks
- Resides at the beginning of the disk
- Status == value (0 free; used otherwise)
- Value of used block == address of next block (last block all bits 1)
- Filenames, attributes&first block address within a directory in the system
- Fixed directories (and files) for FAT-12/16, dynamic with FAT-32



FAT limitations

- Max block nr limited by bits associated to each table entry
  - Bigger files ->bigger blocks->more internal fragmentation
- Difficult cashing for the whole FAT for large disks
- Getting worse with bigger file systems