File Management

- Motivation
  - Common interface to transparently manipulate data on secondary storage
- File system
  - Abstract a storage device (e.g. disk) as
    - A collection of files (data) plus
    - A directory structure (control information)
  - Interaction with storage devices through services exported by the corresponding device drivers
    - Device = linear array of blocks
  - One of the most visible OS structures
  - Examples:
    - FAT, UFS, EXT2, NTFS, ISO9660, etc

Files

- File
  - Named, nonvolatile sequence of bits, bytes, lines or records
- Typed file
  - Internal structure defined by the OS
    - Executable files, graphics files, text files, etc
  - Limited number of known types
- Untyped file
  - Streams of bytes whose meaning is defined by the user
  - Unlimited and flexible

File Attributes

- Name
  - Character string identifying the file to users
- Type (only for typed files)
  - OS internal type information
- Location on device
- Size
- Ownership
- Access control
  - Who can access the file for what operations
- Access history
  - Dates, times, users, counters, etc

File Operations

- Creating
  - Locate space in the file system
  - Create a directory entry
- Deleting
  - Search the directory for the named file
  - Release file system space
  - Remove the corresponding directory entry
- Writing/reading
  - Search the directory for the named file
  - Determine the location in the file system to operate
  - Write/read data
  - Update the file pointer

File Operations

- Positioning
  - Search the directory for the named file
  - Move the file pointer
- Opening/closing
  - Since all file operations require a directory search, it is usual to implement these operations to fetch significant file's information into the system 'table of open files'
- Memory mapping
  - Associate a portion of a process' address space with a section of a file, so that reading and writing to that memory region is equivalent to performing the corresponding operations on the file

File Access Methods

- Sequential
  - Ordered access, one record after the other (tape model)
  - File pointer incremented after each operation
  - Rewind moves the file pointer back to the beginning
- Direct
  - File pointer can be moved arbitrarily (disk model)
- Indexed
  - Based on the direct access method
  - Index associating a search key to records

File Consistency Semantics

- Unix
  - Writes to an open file by a user are immediately visible to all other users that have that file open
  - Locking mechanism for access synchronization
- Session (Andrew)
  - Every new open returns a 'copy' of the file
  - No file concurrency (private copy)
  - Update at close (visible to new sessions)
- Immutable-shared-file (Bullet)
  - Shared files are made read-only

File Access Control

- Motivation
  - Multiuser file system call for access control by the OS
- Types of access
  - Read, write, execute, append, delete
- Access criteria
  - Knowing the name of files
  - Knowing a password associated to files or directories
    - Impractical for interactive applications
  - Being included in a file or directory access list
    - Associate users and access permissions
    - Hard to maintain
    - Variable size structures

File Access Control

- Unix approach
  - Simplified access list
    - Permissions to reading, writing (deleting), and executing (entering)
    - Permissions for owner, owner's group, and others
  - Example

    `drwxr-xr-x dir1` owner can write, all can read and navigate

    `-rwxrwxrwx fil1` all can do everything

    `-r-x------ fil2` owner can execute

    `-r--r--r-- fil3` all can read

Directories

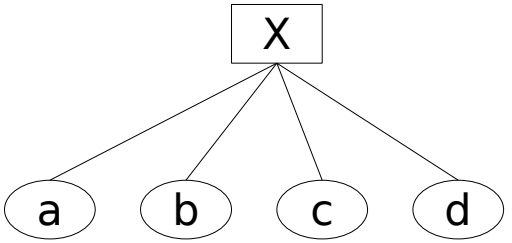
- Directory
  - Collection of information about files
  - Translation table (name => control info)
- Device directory
  - Files' physical characteristics
    - Size, location on disk, owner, etc
- File directory
  - Volume's table of contents
  - Associate file names with device directory entries

Directory Operations

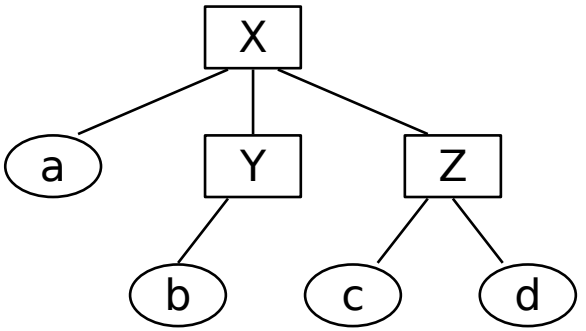
- Create/delete directories
- Add and remove directory entries
  - For file creation/deletion
- Manipulate directory entries
  - For file renaming or control information updating
- Search for a file or pattern
- Listing
- Traversing
  - For file system-wide operations such as search and backup

Directory Organization

- Flat
  - Single directory with all files

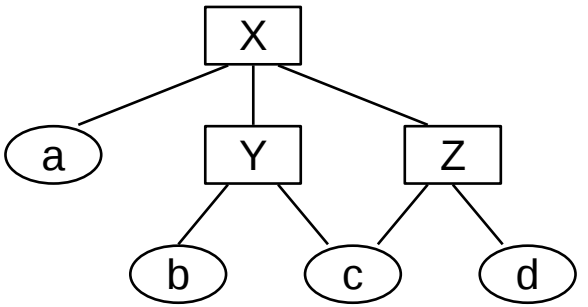


- Tree
  - OS differentiates nodes (directories) and leaves (files)
  - Root node ('/')
  - Pathnames
    - Absolute ('/')
    - Relative to current directory ('CWD')



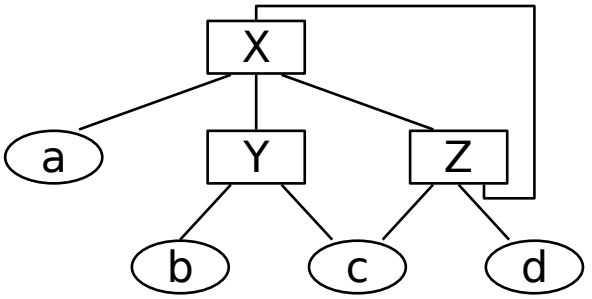
Directory Organization

- Acyclic graph
  - Hard link
    - Reference counter
    - File and link are indistinguishable
    - Not applicable to directories
  - Symbolic link
    - Pathname
    - File and link can be distinguished
    - May become broken
  - Name aliasing problems
    - Deleting
    - Traversing



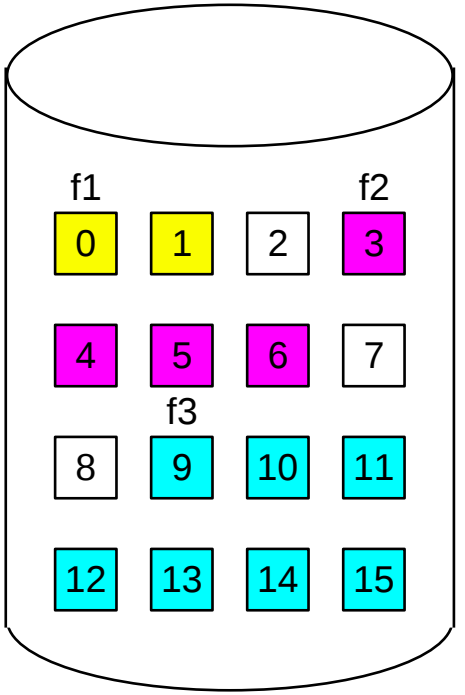
Directory Organization

- General graph
  - Cycles are allowed to exist in the directory
    - Hard links to directories
  - Search algorithm must detect cycles
    - Avoid infinite loops
  - Garbage collection (self reference)



Block Allocation Methods

- Contiguous allocation
  - Directory = (name, start, length)
  - Optimal sequential access plus direct access
  - File size defined at creation-time
    - A sufficiently large set of contiguous blocks must be located (first/best/worst-fit)
  - External fragmentation
    - Garbage collection

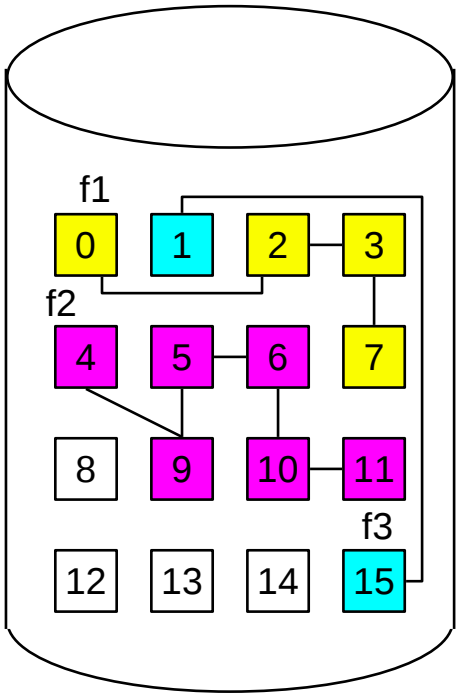


Directory

file	start	length
f1	0	2
f2	3	4
f3	9	7

Block Allocation Methods

- Linked allocation
  - Directory = (name, start, end)
  - Files are linked lists of blocks
    - Any block can be linked to any file
    - No external fragmentation
  - No direct access
  - Limited reliability

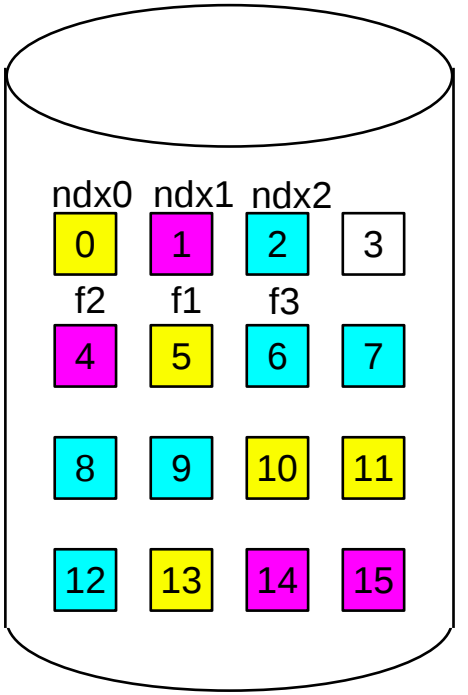


Directory

file	start	end
f1	0	7
f2	4	11
f3	15	1

Block Allocation Methods

- Indexed allocation
  - Directory = (name, index) + index
  - Similar to paging
  - Direct access without external fragmentation
  - Large files
    - Linked index
    - Multilevel index



Directory

file	index
f1	0
f2	1
f3	2

index0
5
10
13
11
-

index1
4
14
15
-
-

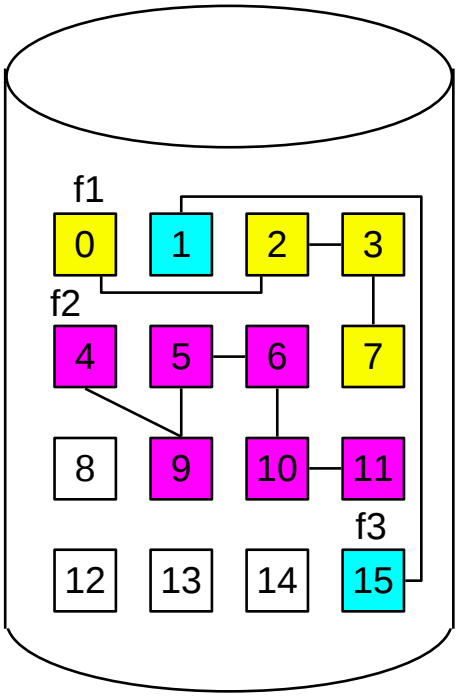
index2
6
7
8
9
12

Free-Block Management

- Bit map
  - Each block is represented by one bit (free/used)
  - Easy to locate sequences of same-state bits
    - Supports contiguous allocations
    - Optimizes sequential access
  - Must reside in memory to be efficient
- Linked list
  - Free blocks are linked in a list
  - Allocation and releasing imply in I/O
- Grouping
  - First free block groups a set of free blocks and contains a pointer to the next grouping block
  - Contiguous ranges of blocks can be represented as pointer + count

Case Study: MS-DOS File Allocation Table (FAT)

- MS-DOS FAT
  - Directory = (name, start, end) + FAT
  - Table with on entry per block is kept separately
  - Special values for free blocks and end of files
  - Allows direct access
  - Reliability improved by replication



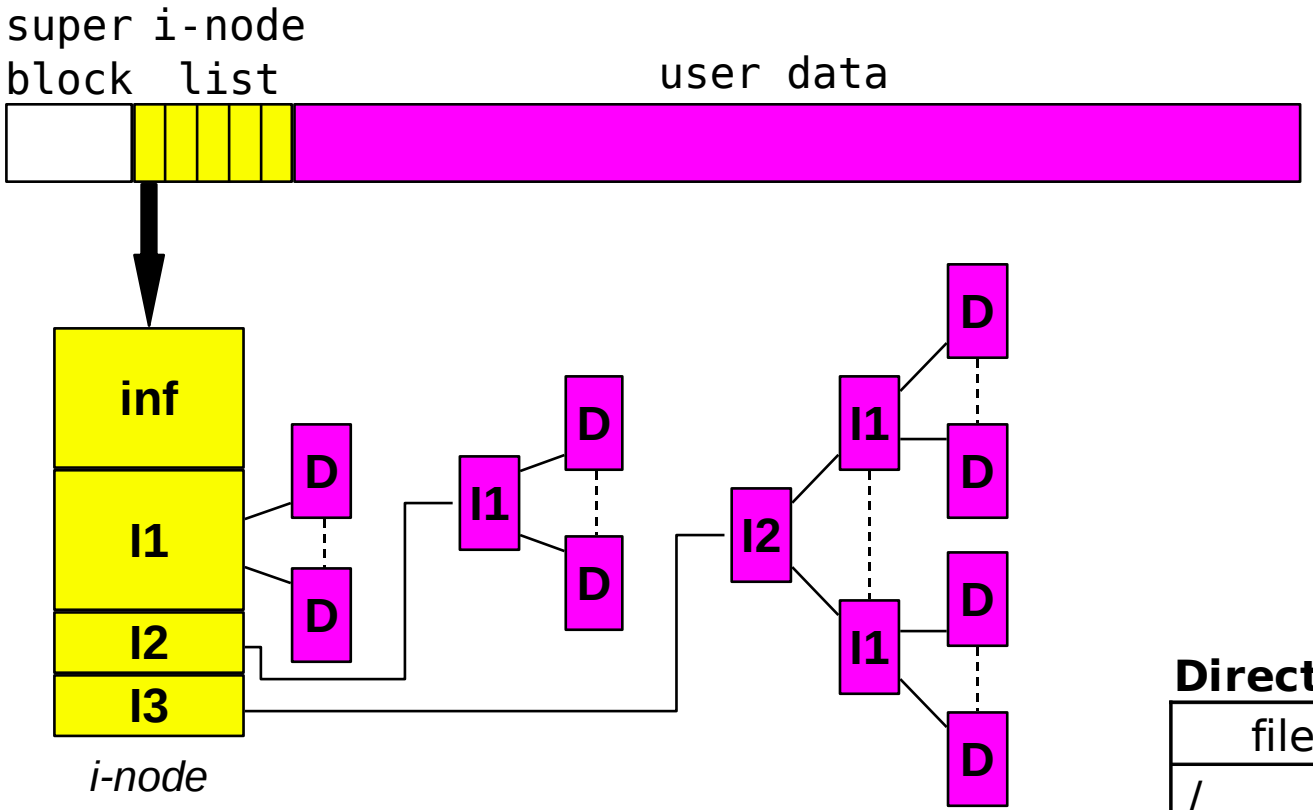
Directory

file	start	end
f1	0	7
f2	4	11
f3	15	1

FAT

2	eof	3	7
9	6	10	eof
free	5	11	eof
free	free	free	1

Case Study: Unix File System



Directory

file	i-node
/	0
f1	1
f2	2