OPERATING SYSTEMS: FILE SYSTEMS

Files

Goals

- To know the concepts of file and directory and their characteristics.
- To use file and directory management services offered by de operating system.
- To understand a file system structure.
- To understand the mechanisms supporting a file server and to apply them to simple exercises.

Content

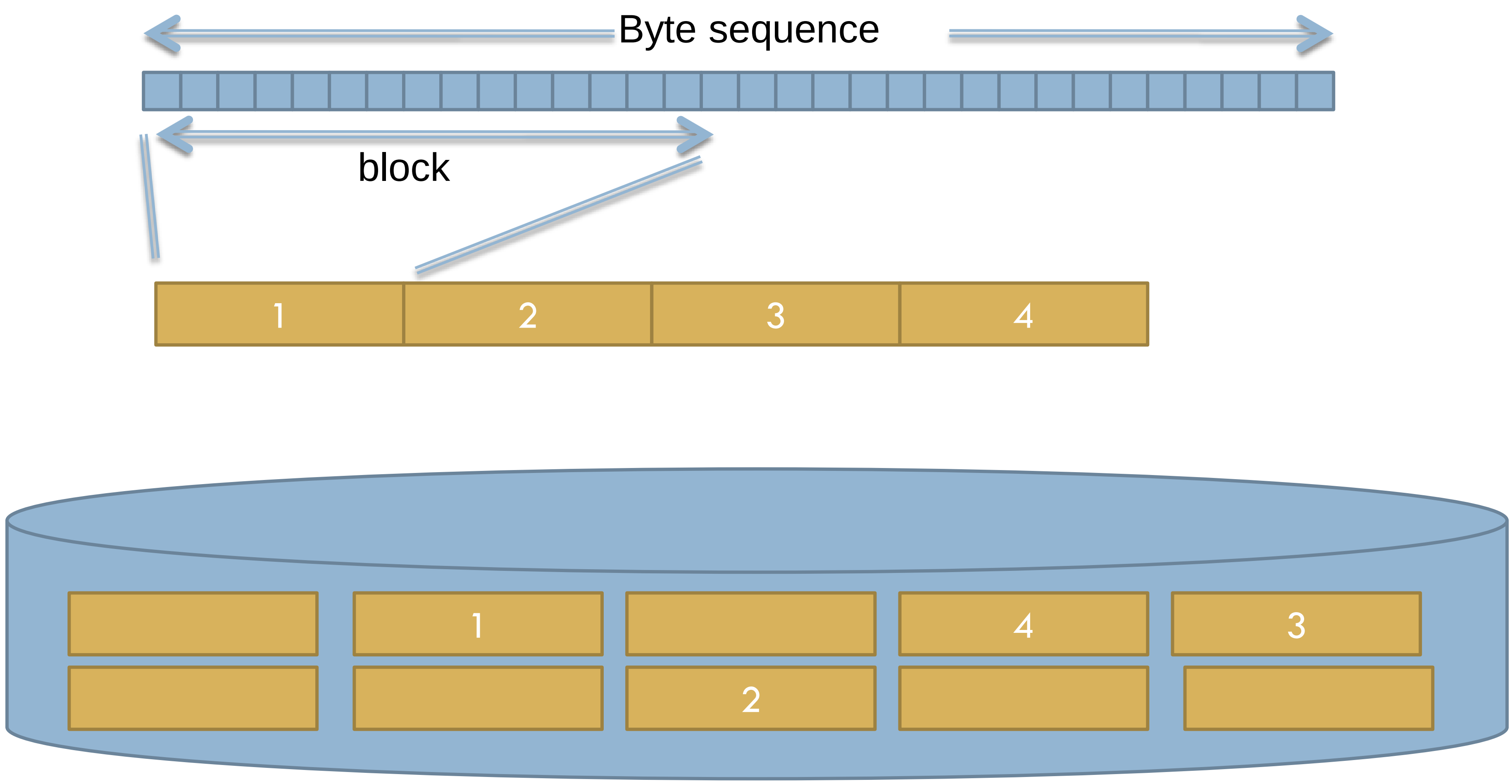
- **Files**
- Attributes and operations
- Logical view
- Sharing semantics
- Representation

Storage

- Main memory
  - Volatile memory → non persistent data.
  - Data accessed directly by processor.

- Secondary memory
  - Non-volatile memory → persistent data.
  - Organized in data blocks.
  - An abstraction needed to simplify applications: **File**.

File

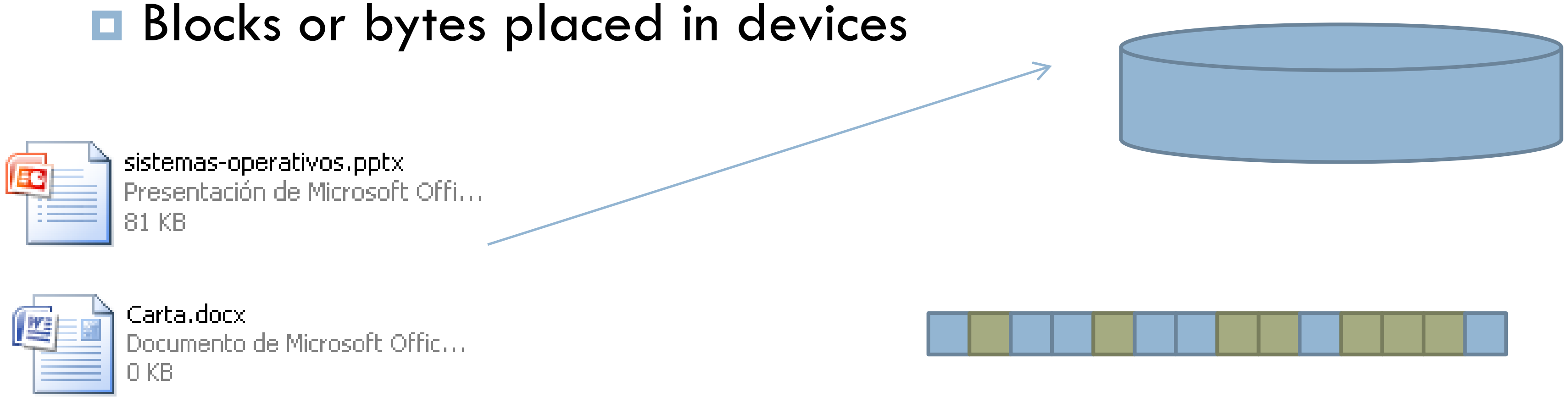


File system

- Offers to user a simplified logical view for handling peripheral devices in form of files.
- Provides an abstraction mechanism hiding details related to storage and information distribution among peripherals.
- The OS subsystem manages files.
- Functions:
  - Organization.
  - Storage.
  - Retrieval.
  - Name management.
  - Implement co-utilization semantics.
  - Protection.

File system: logical view

- Logical view:
  - Files
  - Directories
  - File systems and partitions
- Physical view:
  - Blocks or bytes placed in devices



Features for users

- Permanent storage of information.
  - Does not disappear when computer is switched off.
- Set of information logically structured following application criteria.
- Logical and structured names.
- Dissociated from specific application lifecycle.
- Abstract physical storage devices.
- Accessed through operating system calls or utility libraries.

Content

- Files
- **Attributes and operations**
- Logical view
- Sharing semantics
- Representation

File attributes

- **Name**: Identifier in readable format for a person.
- **Identifier**: Uniquely identifies the file.
  - Usually numeric.
- **File type**: Needed in systems providing multiple file formats.
  - At least used to differentiate executable attribute.
- **Location**: Storage device identification and position in device.
- **File size**: Number of bytes in file, maximum possible size, …
- **Protection**: Access and operations control on file.
- **Time information**: Creation, access, modification, …

File names and extensions

- Characteristic from each file system.
  - Important for users.
- **Problem**: use logical names based in character strings.
- **Motivation**: Users do not remember names like 001223407654

- Type and length change from system to system:
  - Length: fixed in MS-DOS or variable in UNIX, Windows.
  - Extension: Mandatory or not, more than one, fixed per file type, …

- Case sensitive:
  - Example: **SYSTEM** and **system** are the same file in Windows but different on GNU/Linux.

- File system works with internal file descriptors.
  - Only differentiates some formats (executable versus non-executable).
  - Example: magic number in UNIX.

File name and extension

- Directories relate logical names and internal file descriptors.
- Extensions may be used by applications (html, c, cpp, ...)

Name	Size	Type	Modified
My Pictures		File Folder	07/09/2000 11:36
My Webs		File Folder	06/09/2000 11:57
pstr-inf_files		File Folder	14/09/2000 16:21
.fvwmrc	13 KB	FVWMRC File	06/05/1999 18:00
cartacas.tex	1 KB	COREL Texture	06/05/1999 17:55
cata99.ps	193 KB	PS File	06/05/1999 17:55
control.bib	16 KB	BIB File	06/05/1999 17:55
faxing.log	4 KB	Text Document	06/05/1999 17:55
fig3-1.tif	734 KB	Corel PHOTO-PAIN...	22/08/2000 11:59
fig3-7.cdr	27 KB	CDR File	03/05/2000 18:27
pstr-inf.doc	53 KB	Microsoft Word Doc...	14/09/2000 16:21
pstr-inf.htm	1 KB	Microsoft HTML Doc...	14/09/2000 9:51
remain.zip	0 KB	WinZip File	30/05/2000 12:34
Sample.jpg	10 KB	Corel PHOTO-PAIN...	05/09/2000 17:08
winamp265.exe	2.112 KB	Application	07/09/2000 13:10
cmutex.cpp	3 KB	CPP File	11/07/2000 15:30
secobject.c	2 KB	C File	14/07/2000 12:52
adasmspkg.adb	13 KB	ADB File	24/02/2000 9:49
Demo.ppt	345 KB	Microsoft PowerPoi...	24/07/1998 8:15
vol3tc04.html	10 KB	Microsoft HTML Doc...	22/12/1999 11:28
remain.pdf	4.110 KB	Adobe Acrobat Doc...	07/04/1999 11:11

Operations on files

- **Create**: Allocate initial space and metadata.
- **Erase**: Free associated resources.
- **Write**: Store information on file.
- **Read**: Retrieve information from file.

Additional operations depending on concrete file access semantics.

File Systems

- Accessing devices is:
  - **Uncomfortable**:
    - Physical details of devices.
    - Dependent on physical addresses.
  - **Unsafe**:
    - If user accesses to physical level there are no restrictions.
- File System is the software layer between devices and users.
- Goals:
  - To provide a logical view of devices.
  - To offer access primitives easy to use and independent from physical details.
  - Protection mechanisms.

Content

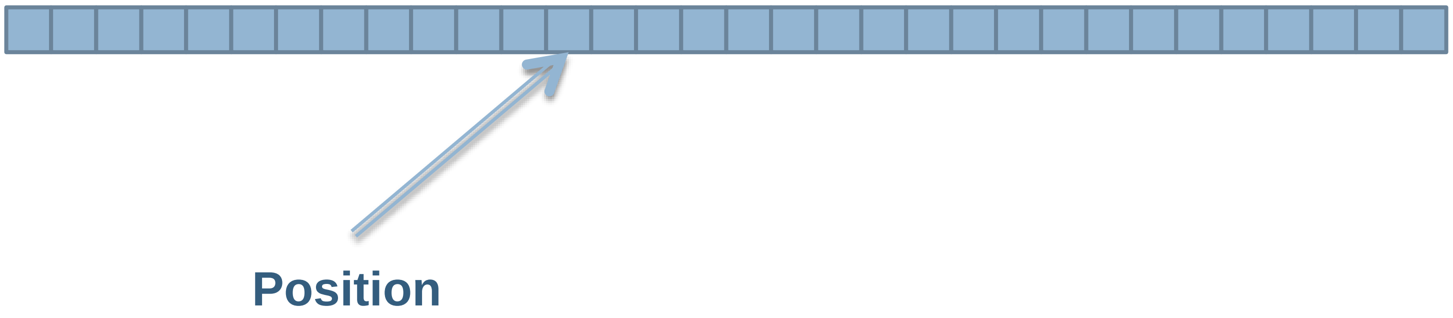
- Files
- Attributes and operations
- **Logical view**
- Sharing semantics
- Representation

File structure

- None: words or bytes sequences (UNIX).
- Simple record structure:
  - Lines.
  - Fixed length.
  - Variable length.
- Complex structures.
  - Formatted documents (HTML, PDF, …)
- Record structures can be simulated on top of a plain structure.
- Who decides on structure?
  - Internal: Operating system.
  - External: Applications.

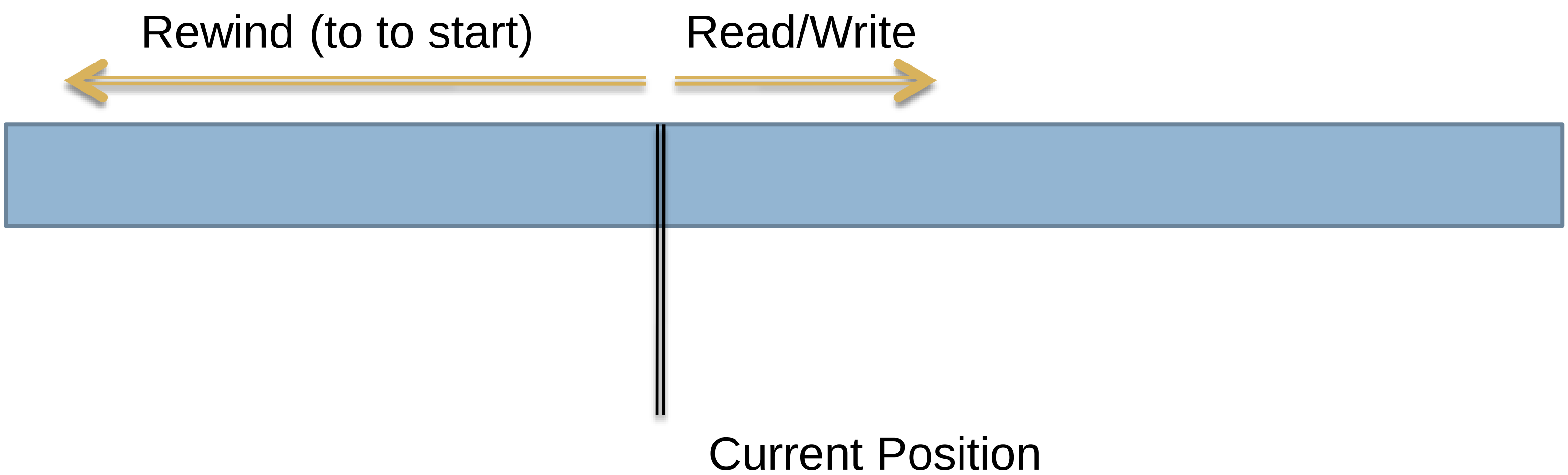
Files: Logical View

- Set of related information that has been defined by creator.
- File structure:
  - Sequence of bytes (UNIX, POSIX)



Access methods

- Sequential access:
  - Based on access model from magnetic tape.
  - Usable in sequential and random-access devices.
  - Byte or record-oriented operations.



Access Methods

- Direct Access
  - Based in access model from disk device.
  - File divided in fixed-length records.
  - May specify record number for read and write operations.
  - May use a position pointer to avoid needing to specify position in every operation.
  - Allows to build on top of it other more complex methods (example: sequential indexed).

Content

- Files
- Attributes and operations
- Logical view
- **Sharing semantics**
- Representation

File sharing

- Several processes may access at the same time to file.

- It is needed to define a coherence semantics.
  - When are modifications to a file observable by other processes.
- Options:
  - UNIX semantic.
  - Session semantic.
  - Immutable files semantic.

UNIX Semantic

- Writes to file are immediately visible to all processes.
- An open file has an associated position pointer.
- Alternatives for the pointer:
  - Each process keeps its own position pointer.
  - Possibility for two processes to share position pointer.
- Implication:
  - Operating system must keep a unique file image.
  - Contention problems due to exclusive access to image.

Session semantic

- Writes on open file are not visible to other processes with that file also open.
- When a file is closed changes are visible to other processes that open the file after that event.
- A file may be associated with several different images.
- There is no contention.
- Use case: AFS (*Andrew File System*).

Immutable semantic

- File may be declared as shared.
  - After that file cannot be modified.

- An immutable file does not admit modifications for:
  - Name.
  - Content.

Version semantics

- Updates performed on copies with version number.
- Only visible when versions are consolidated.
- Explicit synchronization if immediate update is required.

Access control

- Access control lists.
  - Define a list of users of access that can access a file.
  - If there are different access types, then there is one list per access control type.
    - If user is not on the list -> protection violation

- Permissions.
  - Condensed version rwx- rwx- rwx
    - Three access types (rwx).
    - Permissions for three categories (user, group, others).

Content

- Files
- Attributes and operations
- Logical view
- Sharing semantics
- **Representation**

File representation

- Operating system must keep information on files: **metadata.**
- Metadata are file-system dependent.
- Important: An operating system may admit multiple file systems.
  - Example: GNU/Linux my mount partitions in Ext2, NTFS, …
  - Simplification: same access interface (POSIX)

Disk space allocation

- Free/used disk space management.
- Space allocation for each file.

- Aspects:
  - New files: is maximum space allocated on creation?
  - Which allocation unit is used?
  - Which data structure represents file allocation?

Pre-allocation versus dynamic allocation

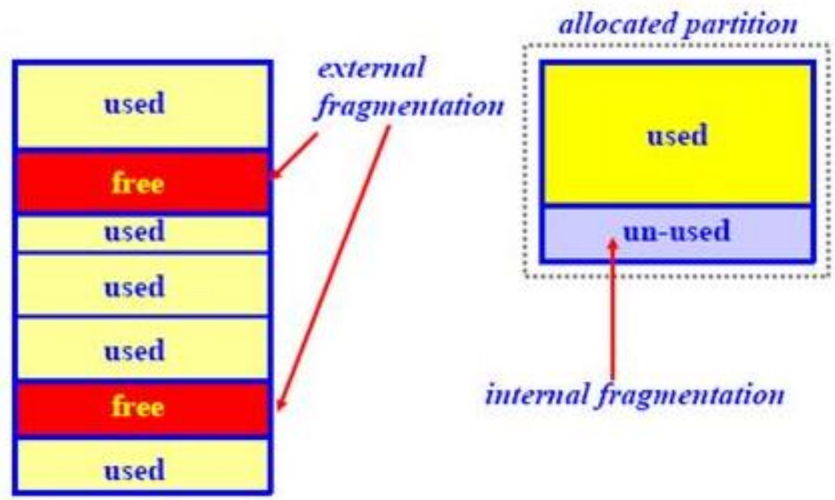
- Pre-allocation:
  - Allocation of maximum possible file size on creation.
  - Maximum space is reserved.

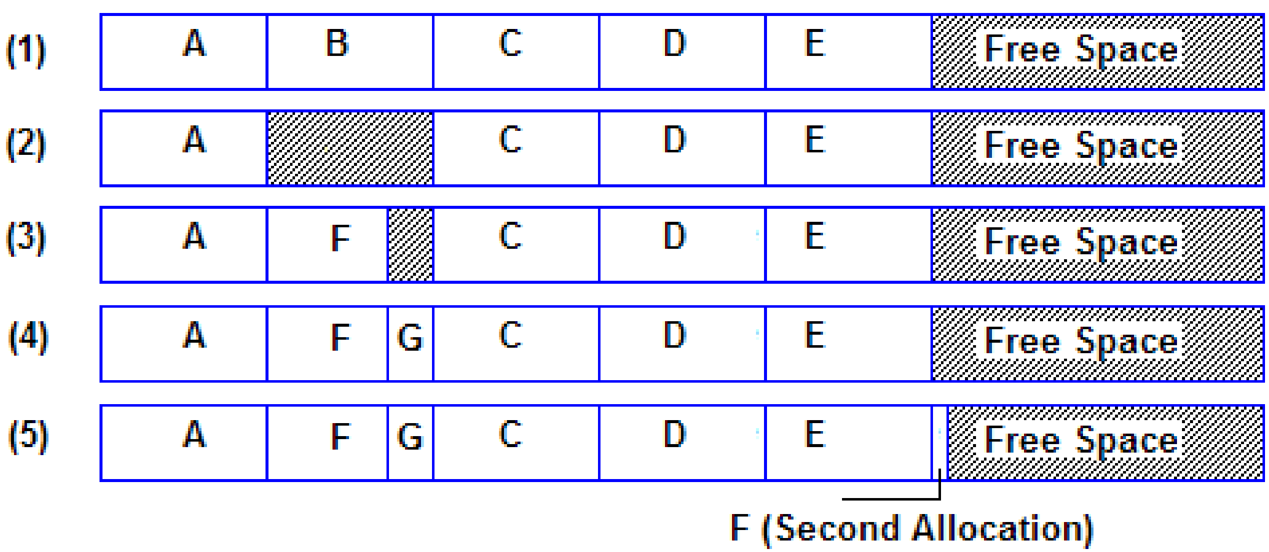
- Dynamic allocation:
  - Space is allocated as it is needed.
  - File divided into allocation units that are taken on demand.

Allocation size

- Issues to be considered:
  - Large allocation size → Information contiguous in disk.
    - Higher performance.
  - Small allocation size → Metadata size increases.
    - Lower storage capacity.
  - Fixed allocation size → Space reallocation is simple.
  - Fixed and large allocation sizes → increments space waste (internal fragmentation).
  - Variable and large allocation size→ increases performance, but external fragmentation increased too.

Fragmentation examples

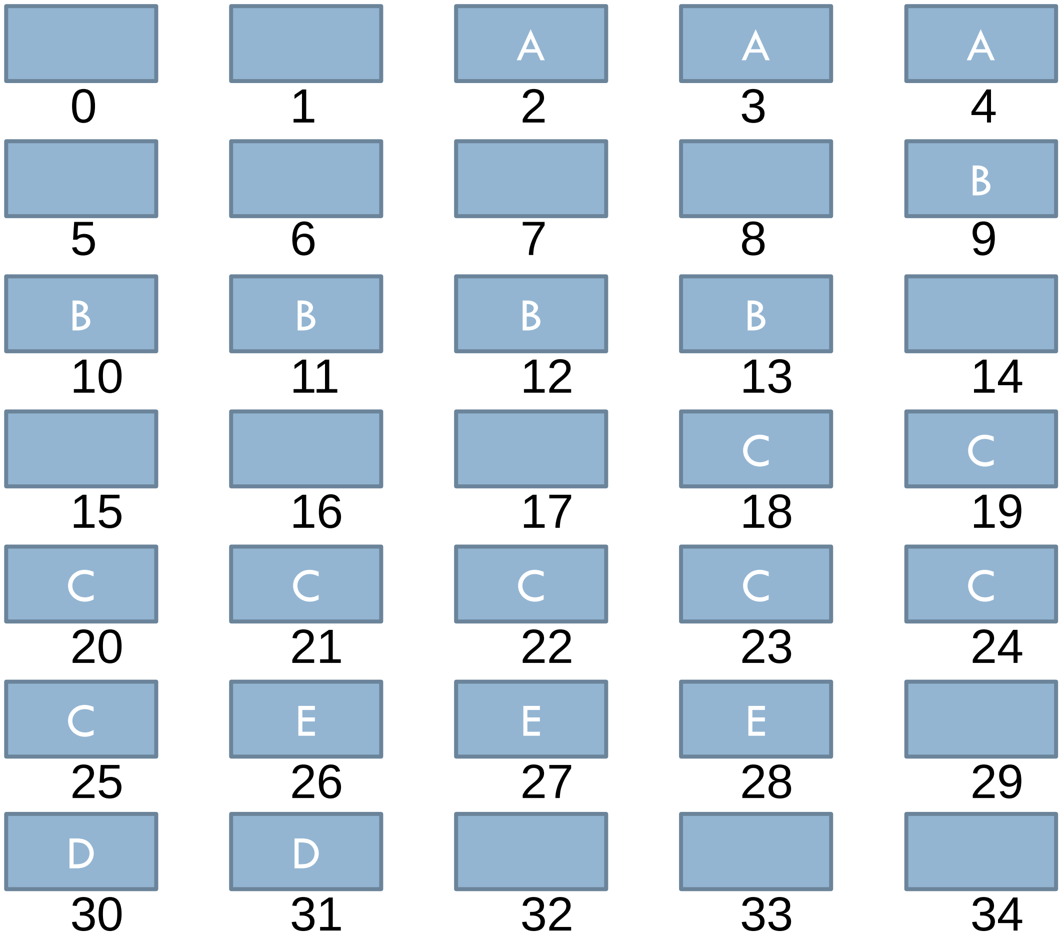




Contiguous Allocation

- 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 easy
- Problems
  - Find space for new file (first/best/worst)
  - Files can not grow more than the allocated space
- Pre-allocation
  - But how much space does a file need?
- Solution: file as a collection of extents
  - A contiguous chunk of blocks
  - When full, add a pointer to the next extent

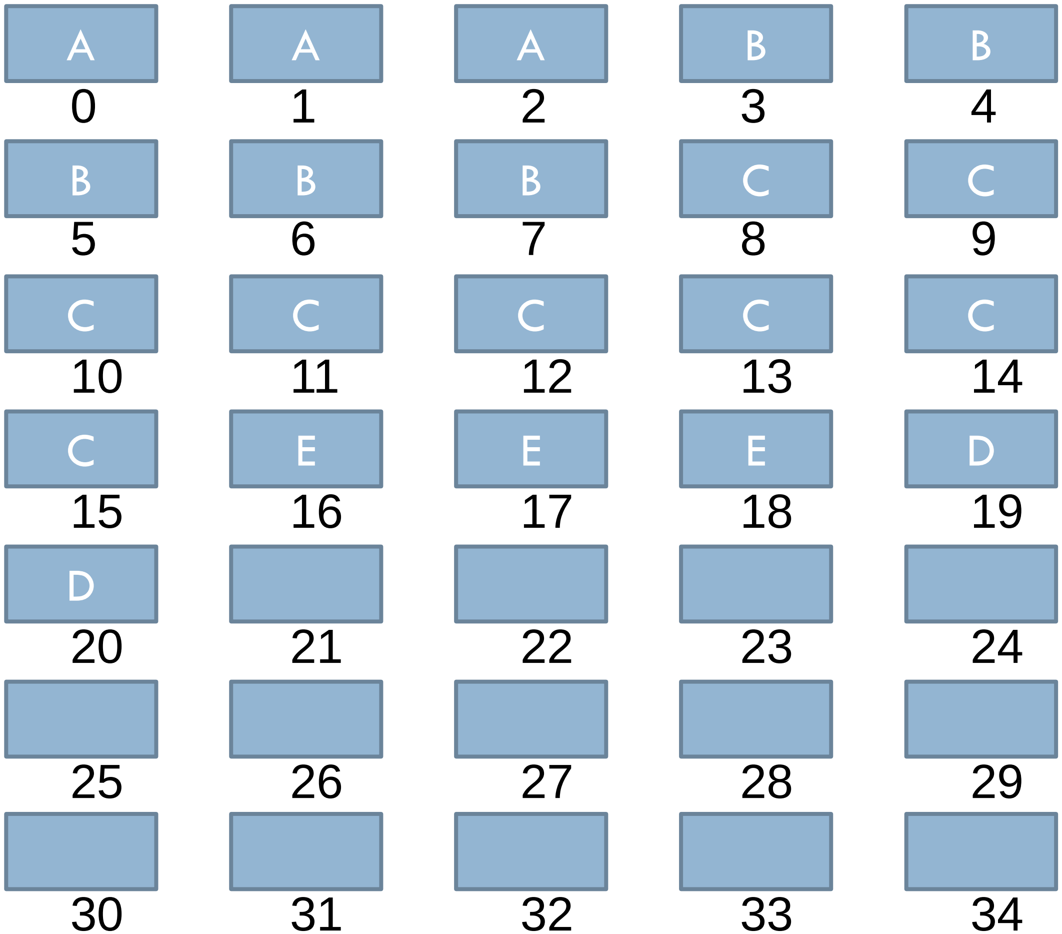
Contiguous allocation



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

Defrag needed

Contiguous allocation (defragmented)

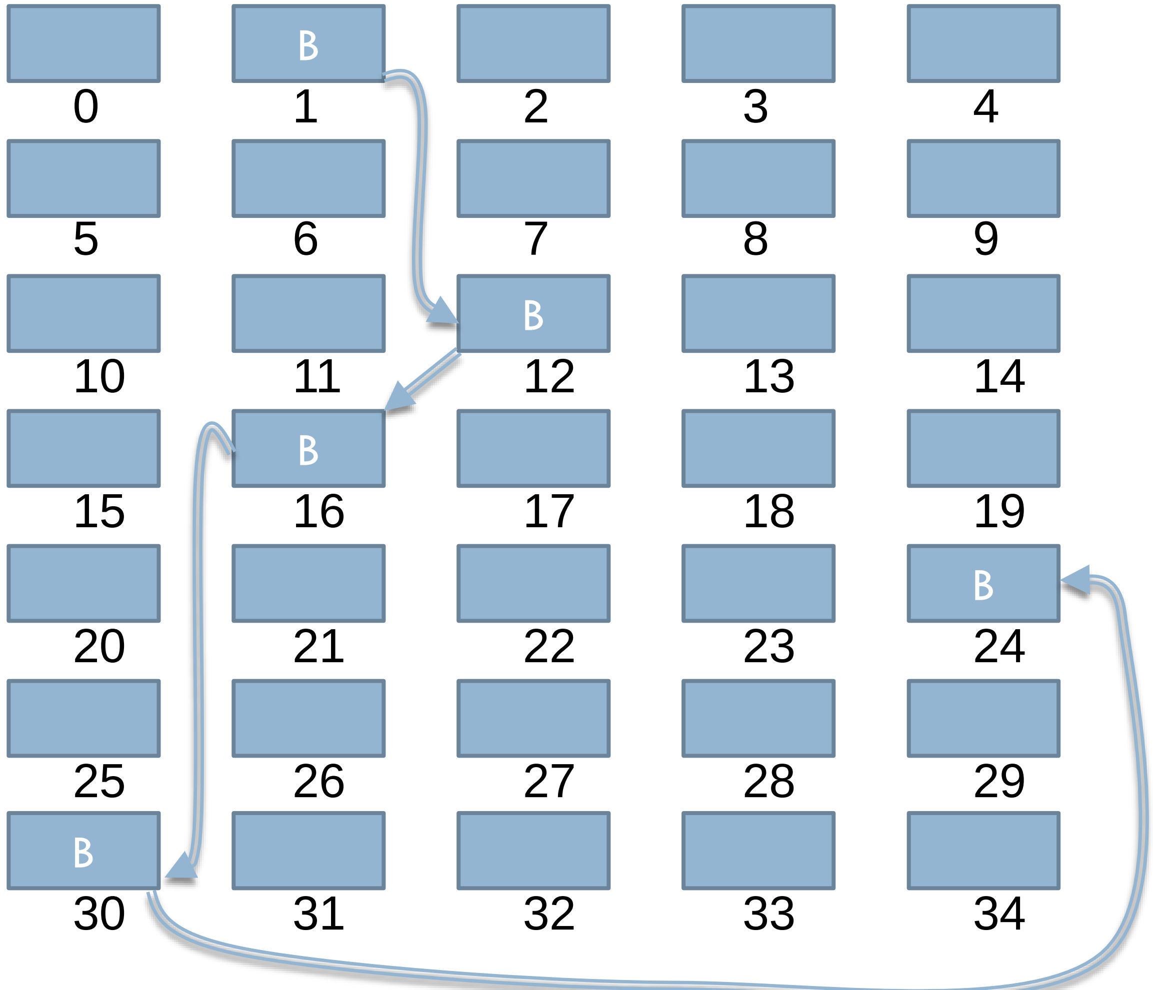


File	Start	Length
A	0	3
B	3	5
C	8	8
D	19	2
E	16	3

Linked allocation

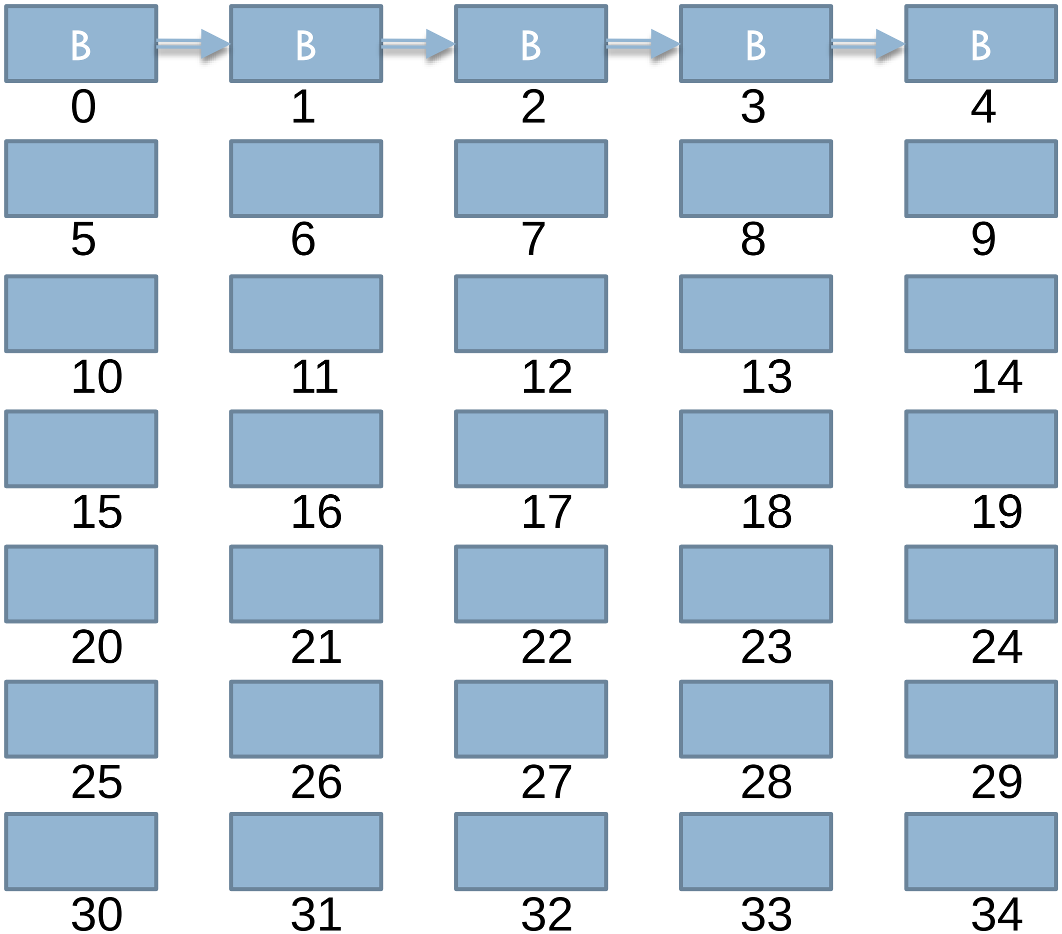
- Each block contains a pointer to next block.
- Block allocation one by one.
- No external fragmentation happens.
- Blocks distributed across disk.
- System consolidation to increase performance in sequential file processing.
  - Increase data locality
- Advantages: Simple – need only starting address, Free-space management system – no waste of space .
- Problems: No random access, Space for pointers.

Linked allocation



File	Start	Length
B	1	5

Linked allocation (consolidated)



File	Start	Length
B	0	5

Indexed allocation

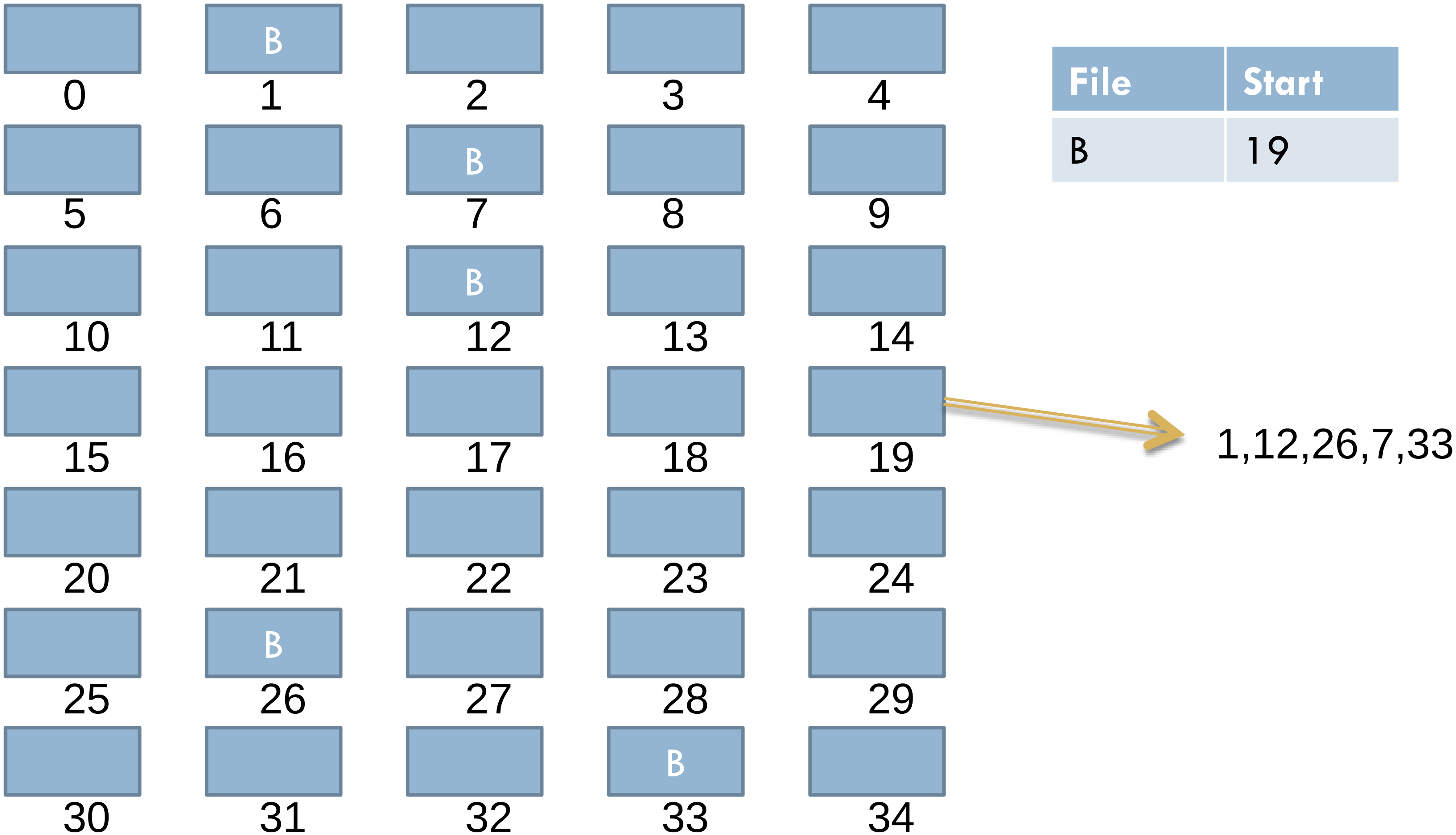
- Table with allocation units identifiers composing the file.
- Alternatives:
  - Block allocation.
  - Extent allocation.

Indexed Allocation

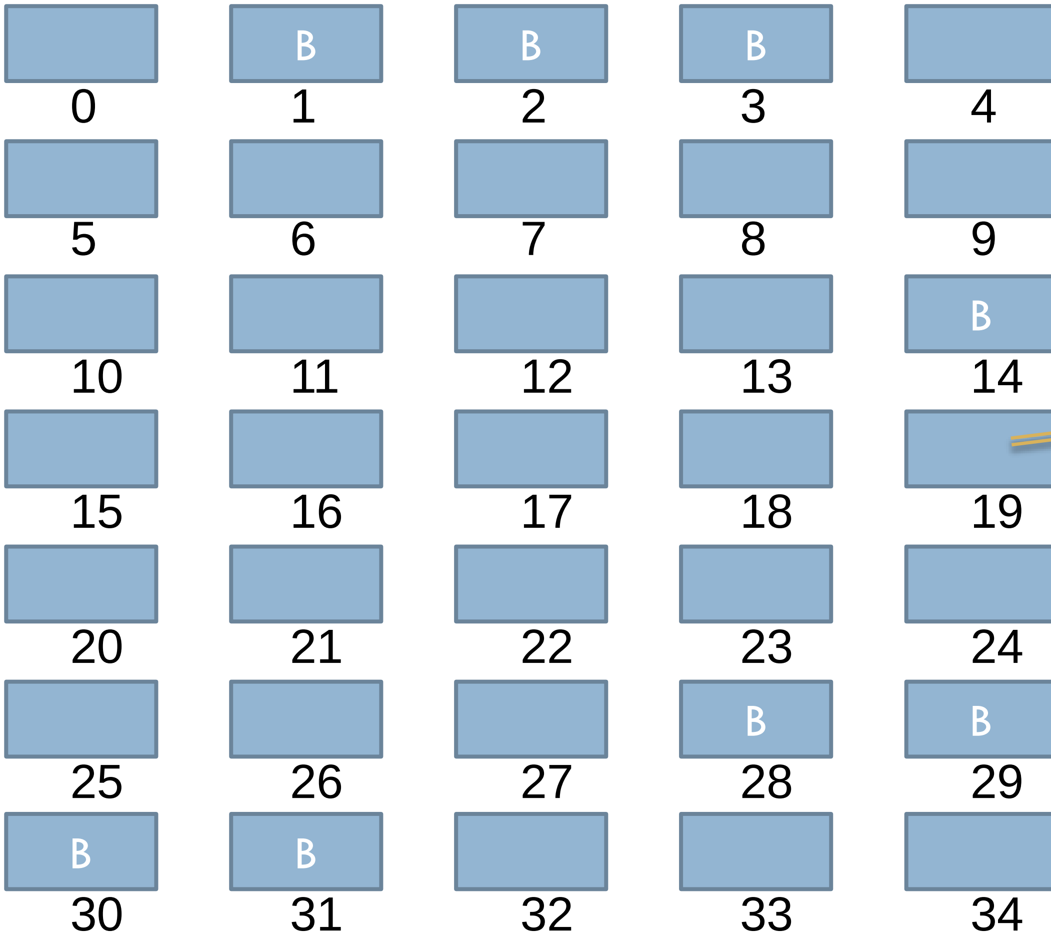
- Advantages
  - Random access
  - Dynamic access without external fragmentation
- Problem
  - Overhead of index block.
- Index block organization
  - Linked scheme: an index block is one disk block containing pointers to data blocks
  - Multilevel index: ex. a disk block contains pointers to blocks containing pointers to data blocks
  - Combined
    - Both (UNIX)

Block indexed allocation

File	Start
B	19



Extent allocation



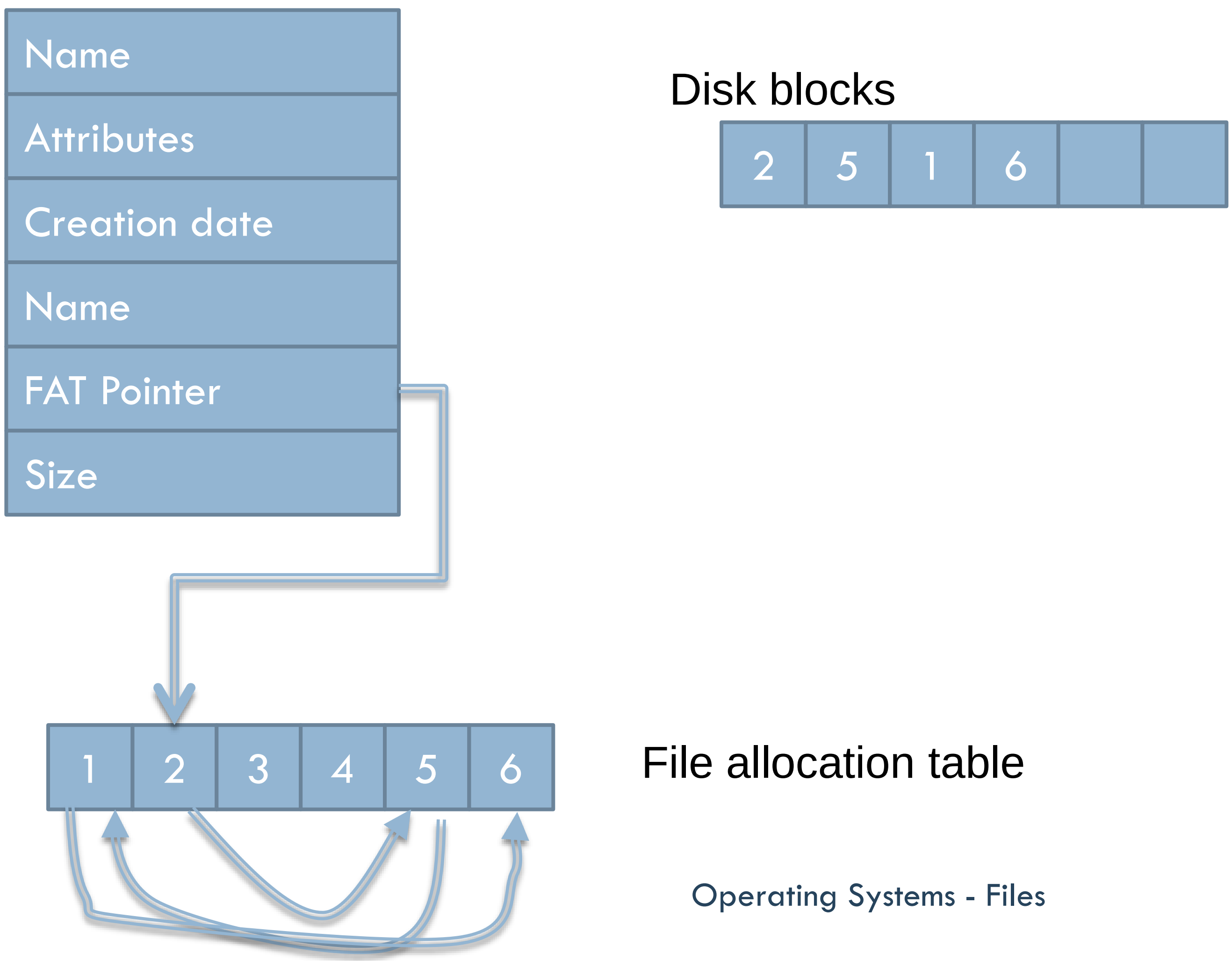
File	Start
B	19

Start	Length
1	3
28	4
14	1

Disk space management

- Operating system must know which blocks are free or used.
- Alternatives:
  - Bitmaps: Vector with one bit per block
    - Summary table with address ranges: Number of free blocks in range.
  - Linked list of free extents.
  - Indexing: index table of free extents.

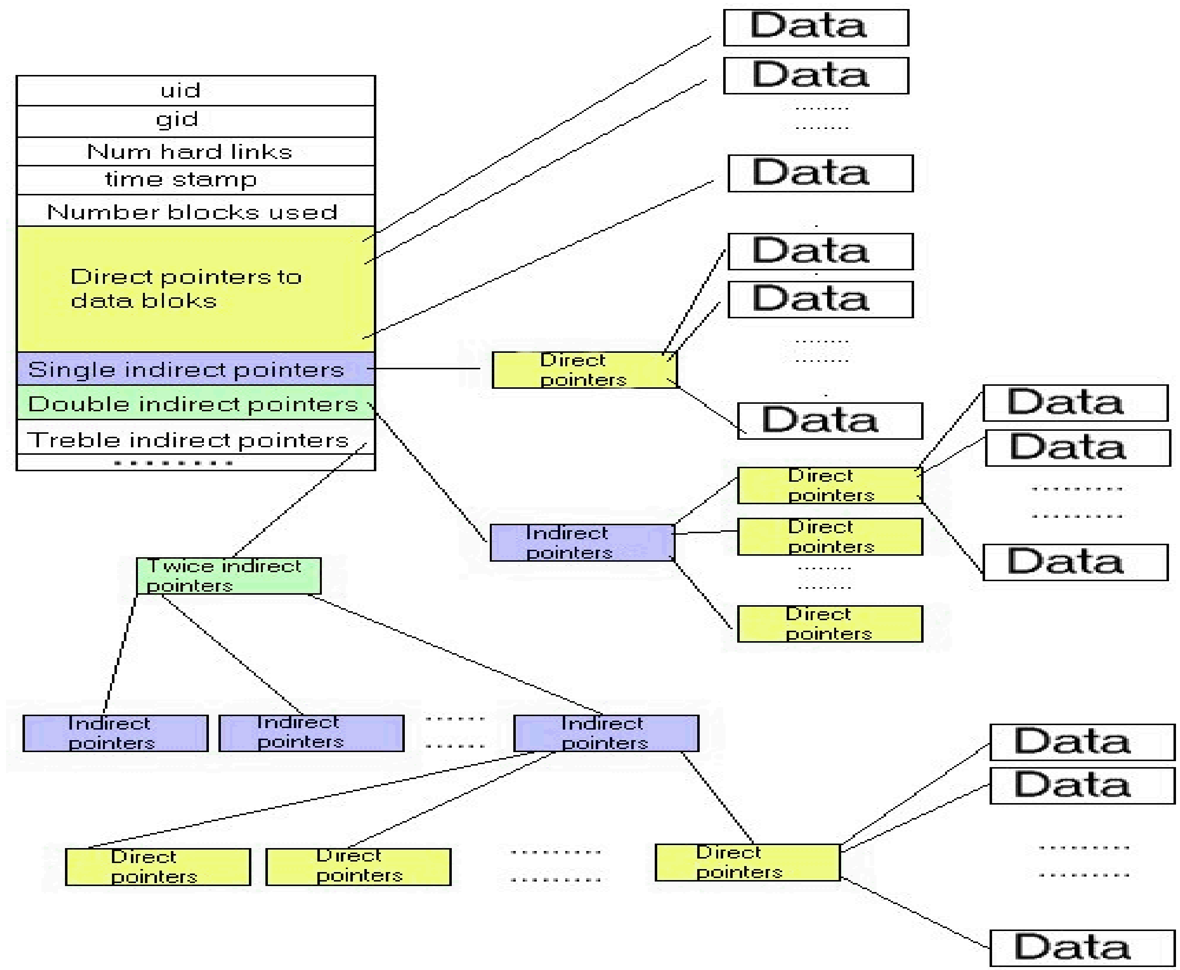
Representation: FAT



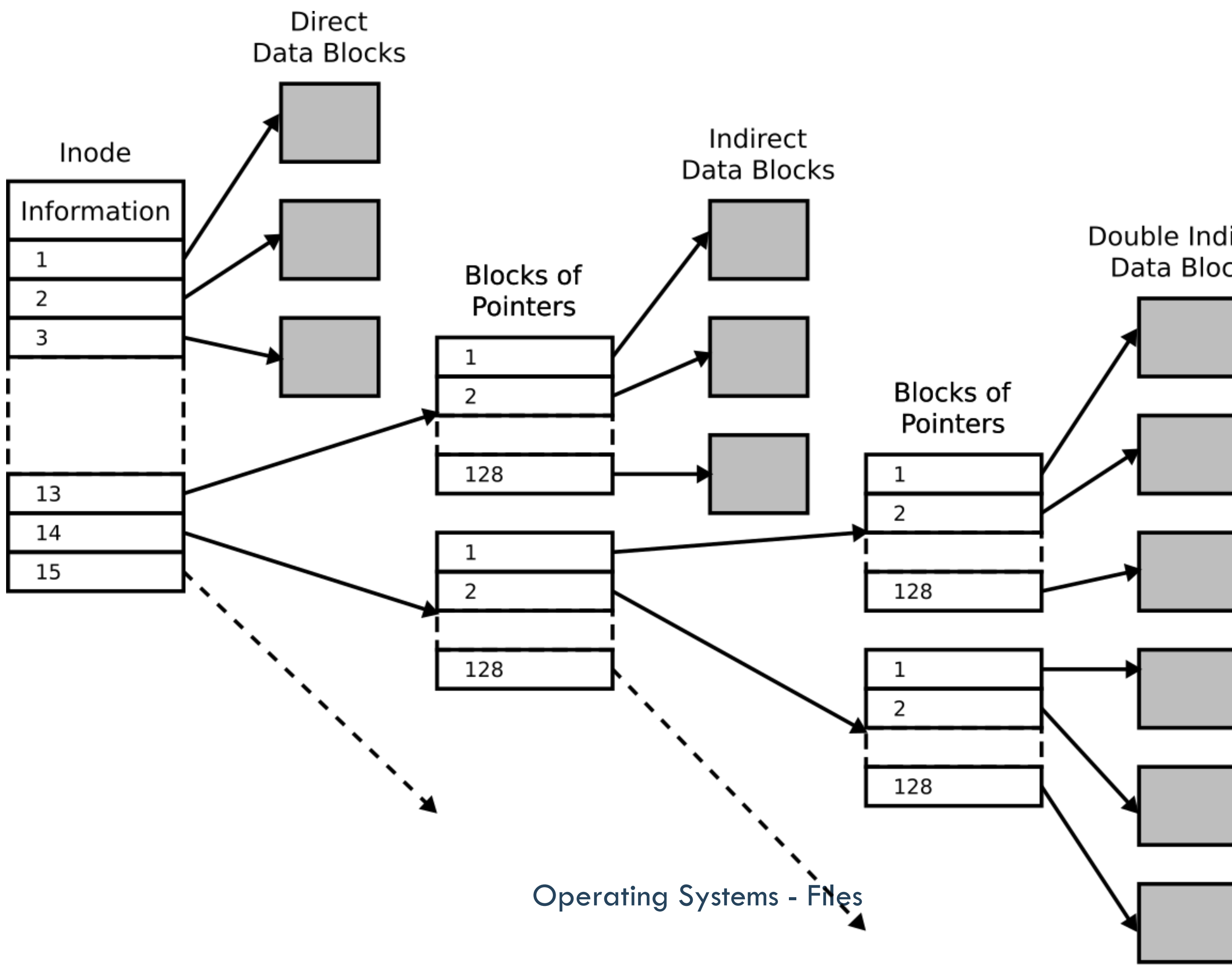
Use case: UNIX

- File type and protection.
- File owner user.
- File owner group.
- File size.
- Creation date and time.
- Last access date and time.
- Last modification date and time.
- Number of links.
- Direct block pointers (10).
- Simple indirect pointer.
- Double indirect pointer.
- Triple indirect pointer.

UNIX: Block pointers



UNIX: Block pointer example



Representation: NTFS