OPERATING SYSTEMS

CSCI 509

FILE SYSTEMS

- Purpose: Abstraction of storage.
  - Programmers and users want to deal with files not with blocks of data.
- Services:
  - File abstraction: Physically, a file is just a bunch of bits on sectors that might be spread all over the disk.
  - File Manipulation: create, write, append, copy, delete …
  - File protection: ownerships, read/write/execute privileges.

FILE ATTRIBUTES

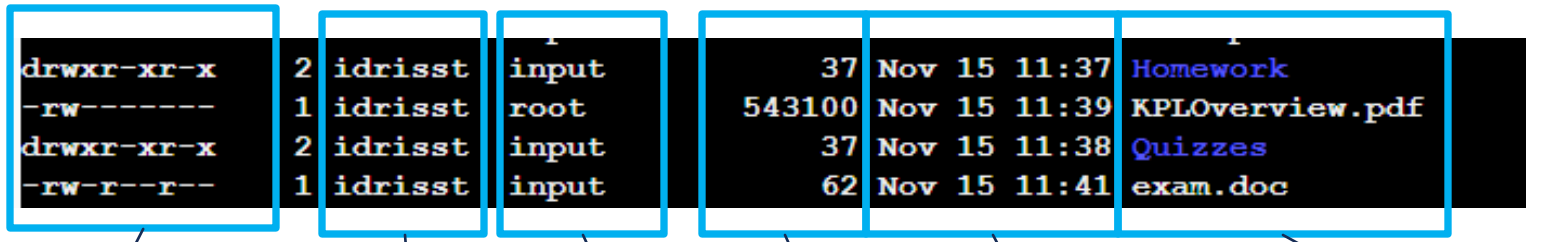
Q: What file information does the File System keep on record?

- **Name** – must be kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring

FILE ATTRIBUTES

- Linux command to view file attributes?

```
ls -al
```



read/write access

owner

group

size

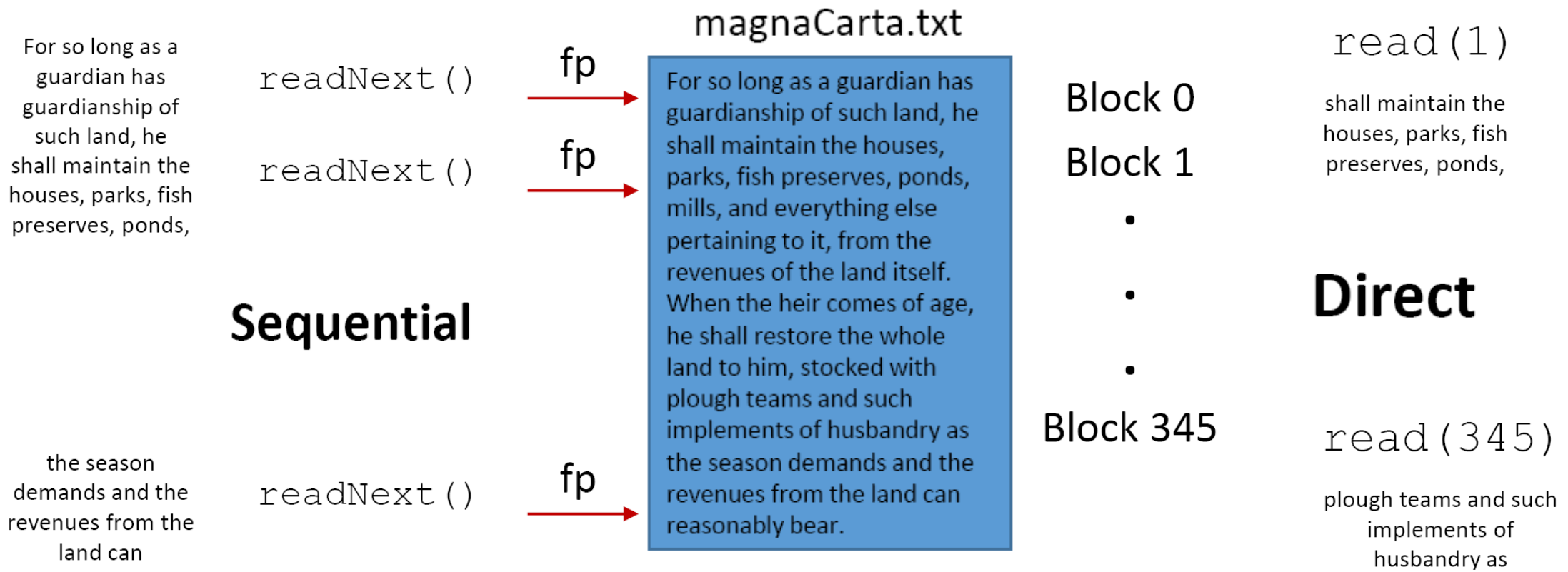
Last modified

File name

FILE ACCESS

- Two methods for file access:
  - Sequential
  - Direct

FILE ACCESS



writeNext()

Writing to the file happens pretty much the same way (sequential vs direct)

writeTo(n)

FILE ACCESS

Sequential

- **Advantages:**
  - Simple to implement.
  - You only need to track the next read/write with a pointer.
- **Disadvantage:**
  - You can only access next address.

magnaCarta.txt

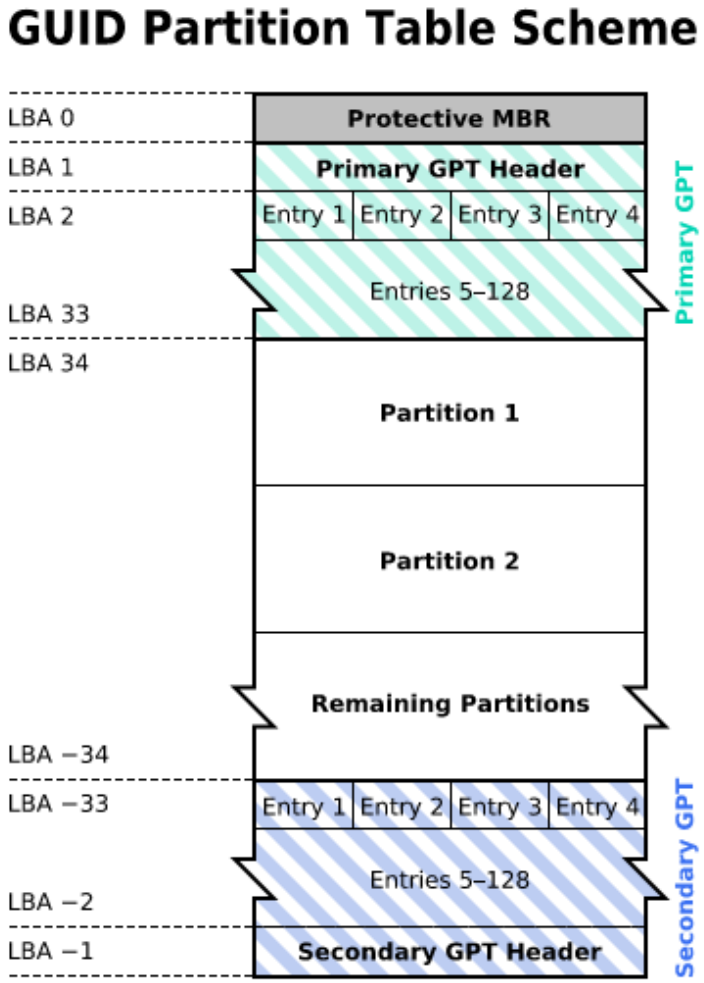
For so long as a guardian has guardianship of such land, he shall maintain the houses, parks, fish preserves, ponds, mills, and everything else pertaining to it, from the revenues of the land itself. When the heir comes of age, he shall restore the whole land to him, stocked with plough teams and such implements of husbandry as the season demands and the revenues from the land can reasonably bear.

Direct

- **Advantage:**
  - Can access any block of the file directly. Much faster than going through unnecessary parts.
- **Disadvantage:**
  - More complex implementation.
  - File system needs to keep more information about the file.
  - Logical address translation table of all blocks should be kept in memory.

DISK PARTITIONING

- How is the file system implemented on disk?
- Raw disk needs to be 'formatted' according to the file system.
- The process is also called partitioning, which is a high-level formatting.
- A file system 'volume' is created. Data structures are stored on disk to keep tracks of files and partitions created.

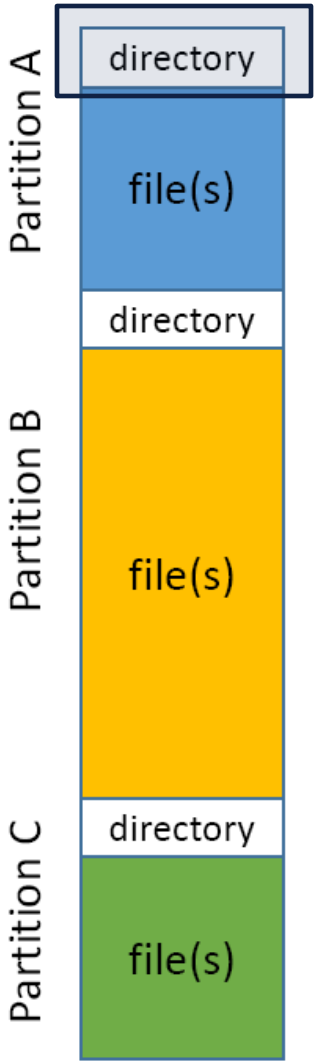


DISK PARTITIONING

Q: Why would we want to split a disk into different partitions?

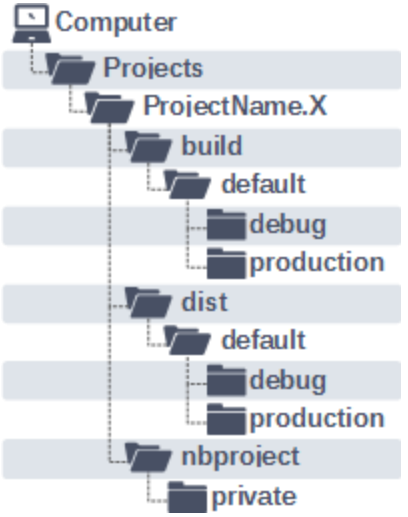
Advantages:

- Different Partitions have their own file system and directory structure.
- Some can be kept "raw" and be reserved for OS for fast access.
- Smaller partitions would mean smaller directory structure and faster walk through.



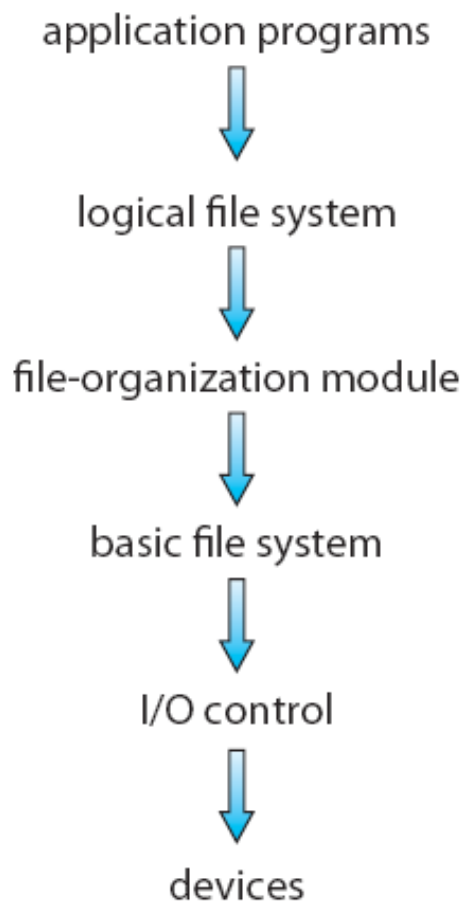
Device directory : retains details about the files in each volume

- What are the files present in each partition?
- Attributes of each file.
- Location or "path" in the directory structure.
- Location of each file on disk.



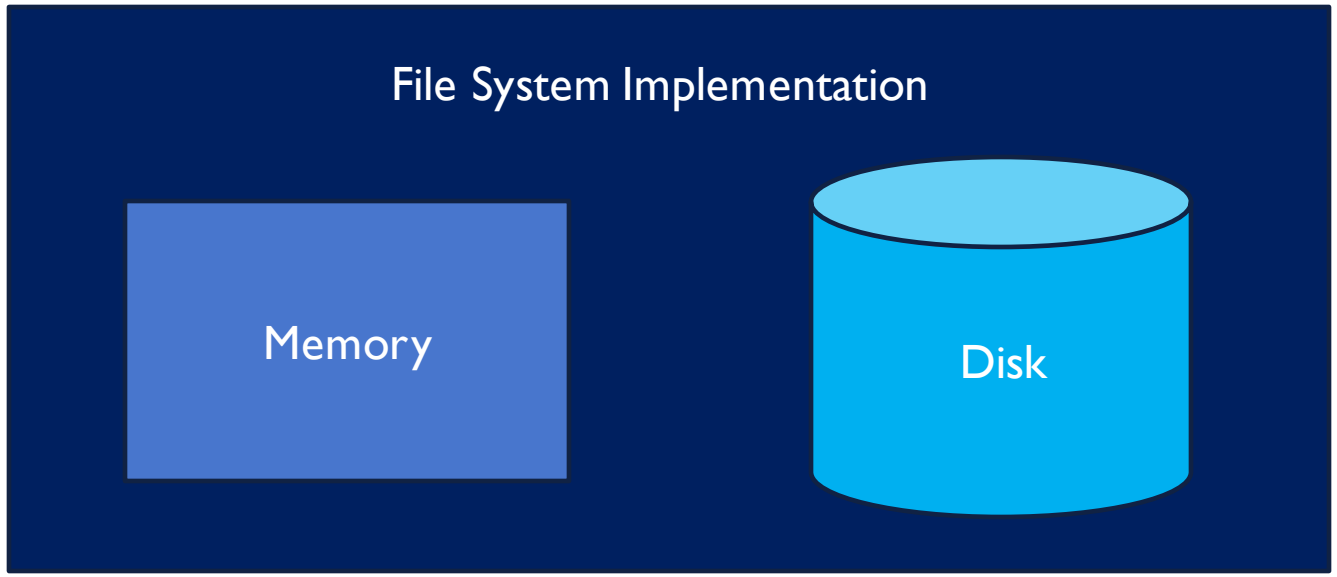
LAYERED FILE SYSTEM IMPLEMENTATION

application	Make the high-level request … get “this” file
Logical file system	Directory and file metadata, including **File Control Block** (**FCB**) [inode in Unix], which caches file ownership, permissions, location, etc.
File organization module	Mapping of file names and/or IDs and their logical block location, as well as free space manager
Basic file system	High-level commands such as “read block 32”
I/O Control	Drivers, interrupt handlers … which location (sector, cylinder) needs to be read from or written to
File system device (most often HD)	The physical device

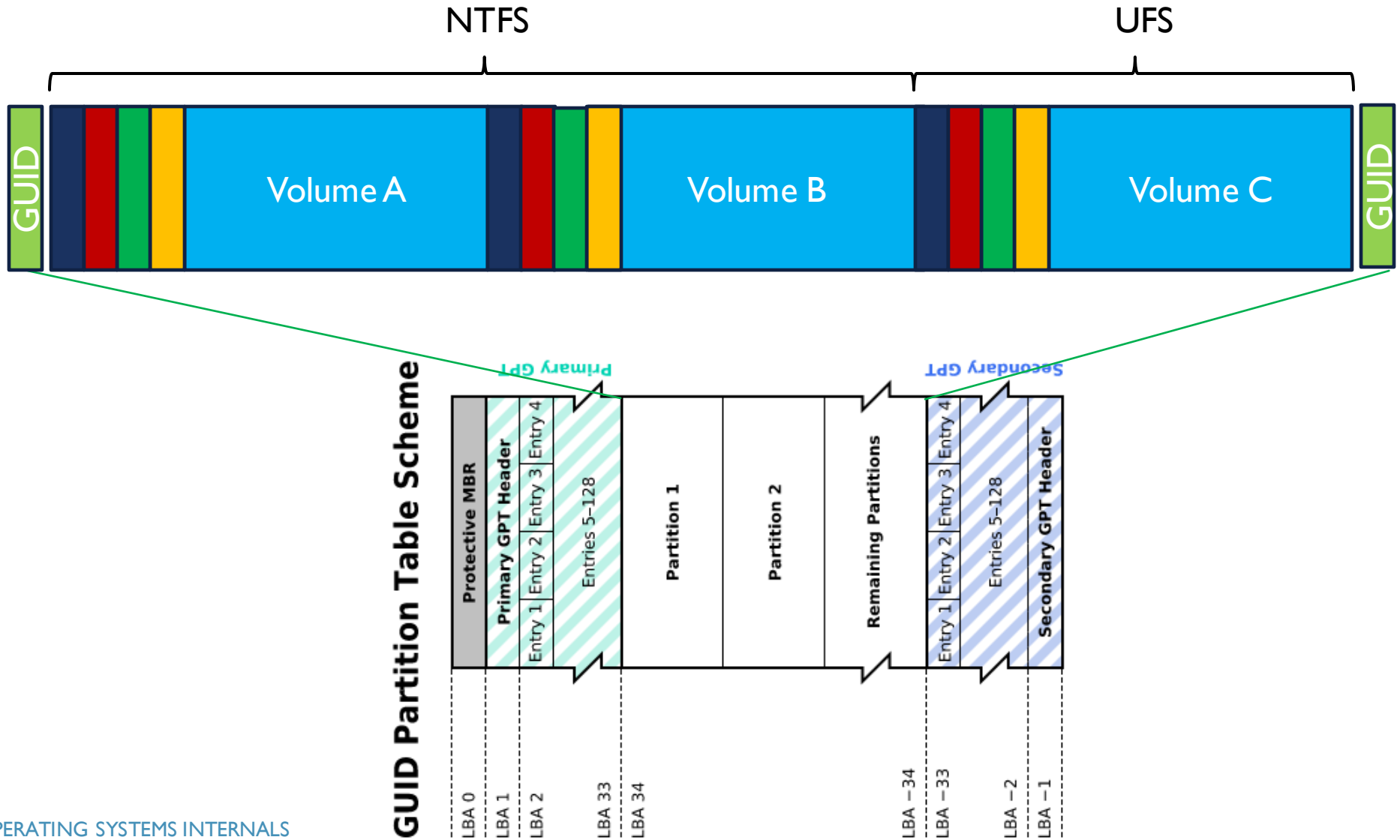


FILE SYSTEM IMPLEMENTATION

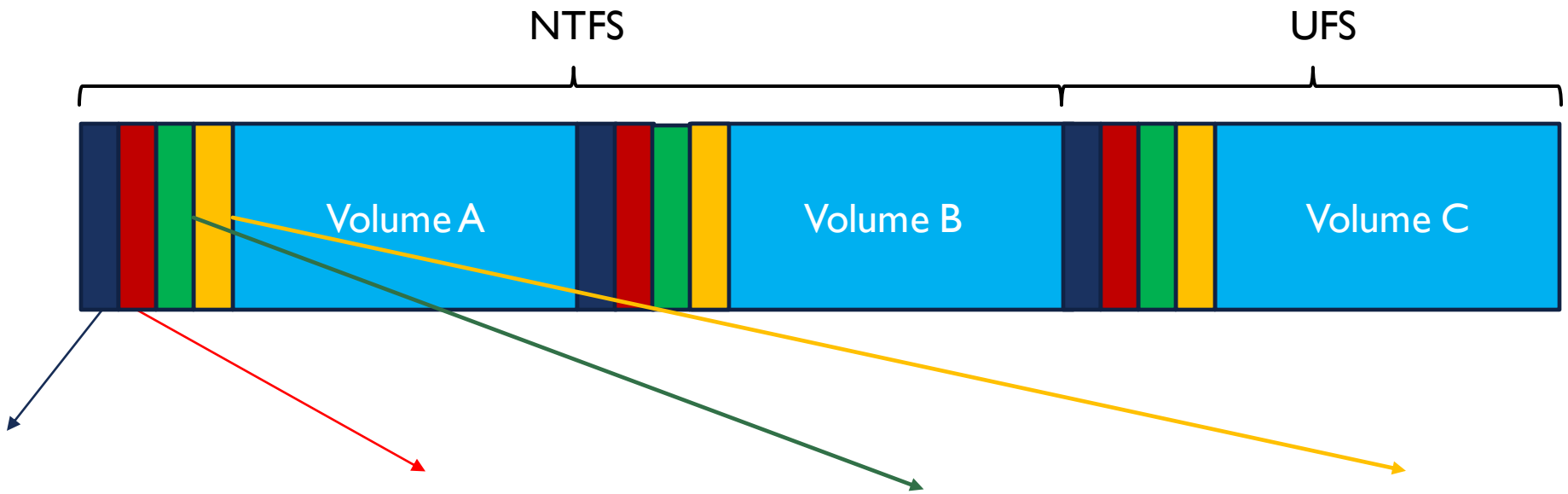
- In Memory
- On Disk/Storage



ON DISK FILE SYSTEM IMPLEMENTATION



ON DISK FILE SYSTEM IMPLEMENTATION



Boot Control Block:

- Information needed by the system to boot the operating system.
- Can be empty if there is no OS on the volume

Volume Control Block:

- Number of blocks
- Number of free blocks
- Free blocks location/pointer

Directory Structure:

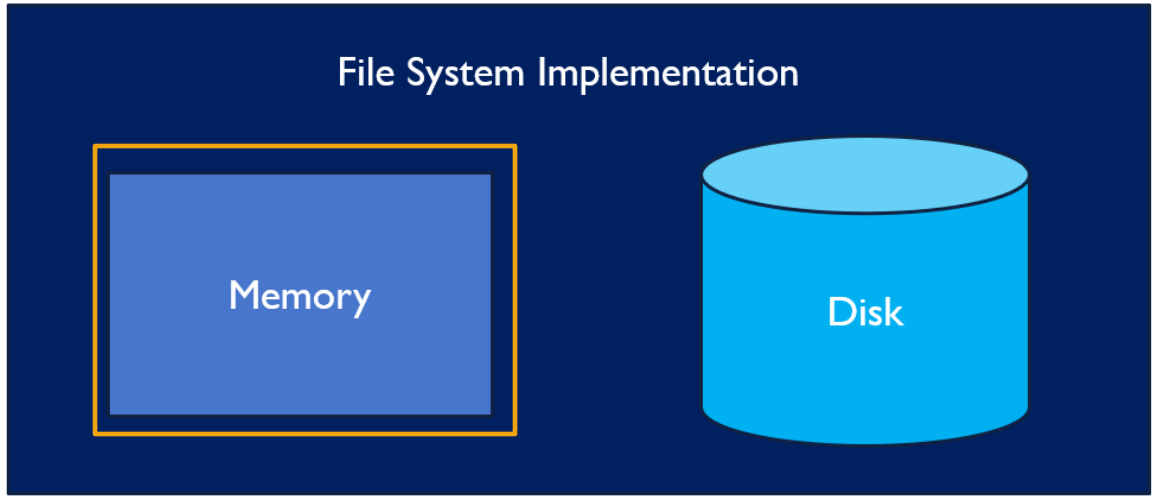
- File organization and names.
- One per file system on device.

File Control Block:

- Per file
- Information about each file
- In NTFS, a Master File Table for all files is used.
- In Linux, Inodes are used.

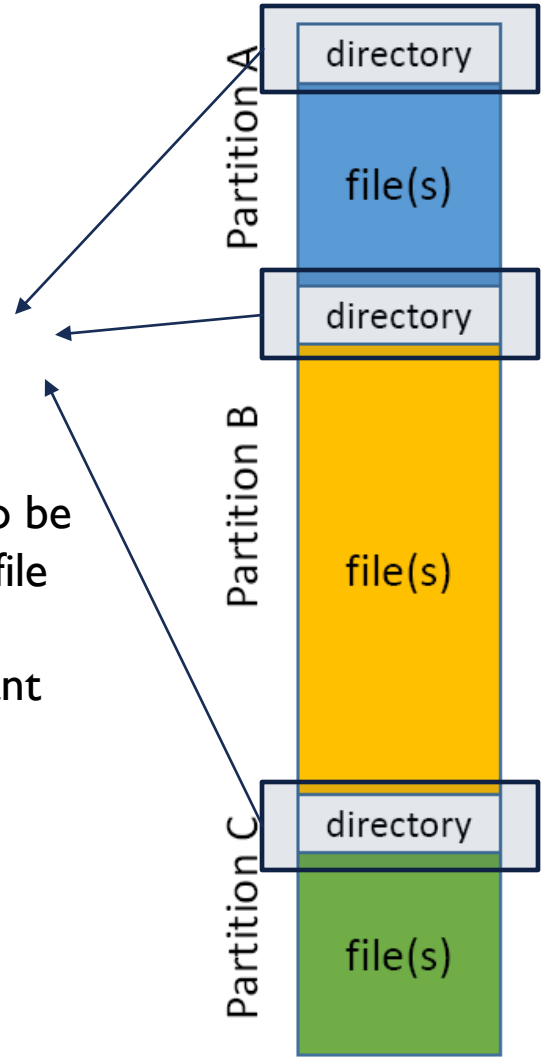
IN MEMORY FILE SYSTEM IMPLEMENTATION

- **Mount Table** contains information about each mounted volume.
  - Mounting a "volume" or a file system is essential before any access.
  - It can be done automatically at boot or using explicit commands.



IN MEMORY FILE SYSTEM IMPLEMENTATION

- **Mount Table** contains information about each mounted volume.
- **Directory Structure Cache** holds the directory information of recently accessed directories. (For directories at which volumes are mounted, it can contain a pointer to the volume table.)
  - This is different than dedicated caching (ex. TLB)
  - By "caching" we only mean we keep portion of it in main memory. There is no dedicated hardware for directory caching.



- Directory needs to be searched for each file access.
- Caching is important for speedup.

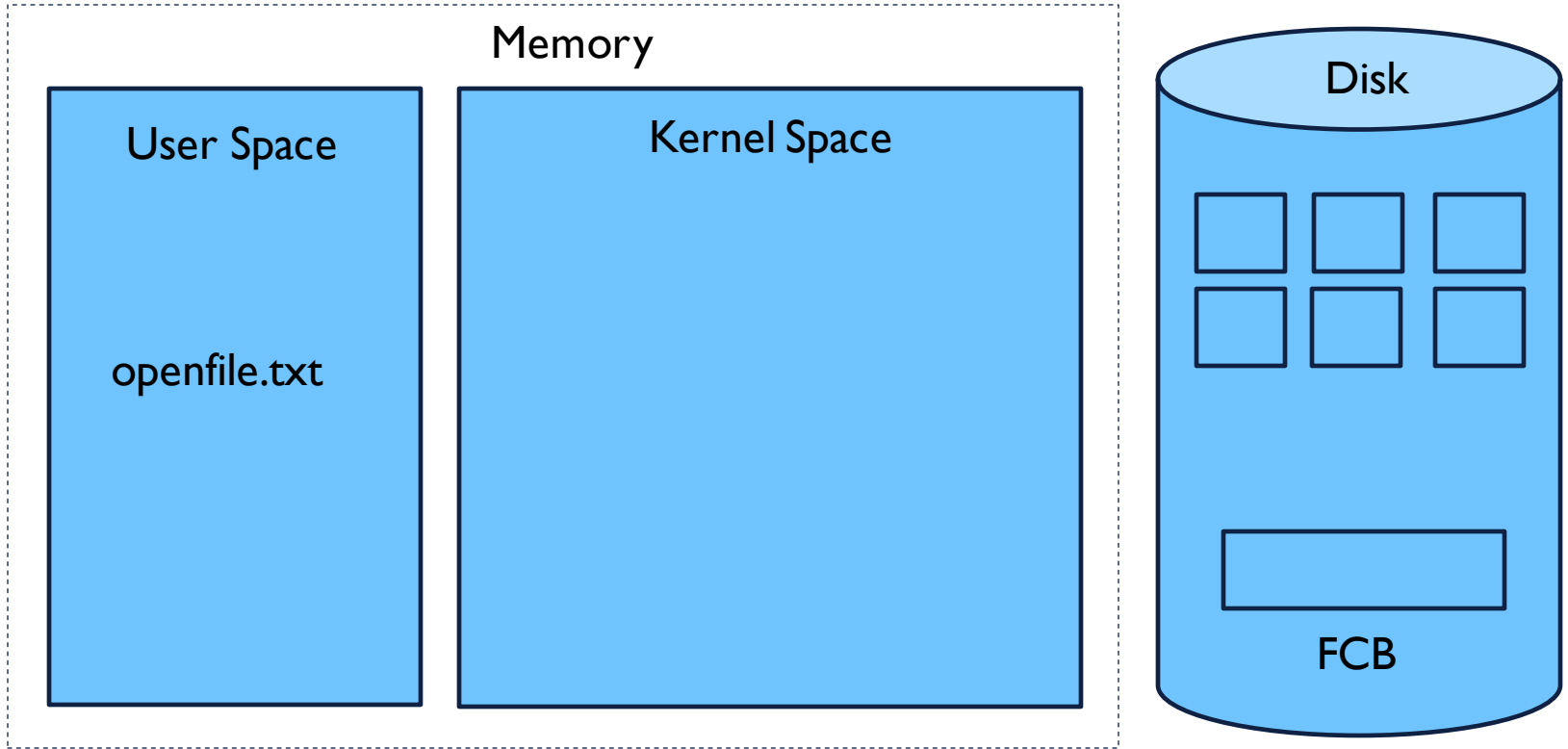
IN MEMORY FILE SYSTEM IMPLEMENTATION

- **Mount Table** contains information about each mounted volume.
- **Directory Structure Cache** holds the directory information of recently accessed directories. (For directories at which volumes are mounted, it can contain a pointer to the volume table.)
- **System-wide Open File Table** contains a copy of the FCB of all open files, as well as other information.
- **Per-Process Open-file Table** contains pointers to the appropriate entries in the system-wide open-file table, as well as other information, for all files the process has open.
- **Buffers** hold file-system blocks when they are being read from or written to a file system.

- No need to write back to disk on every modification.
- Flag file as "modified" and write back on demand.

IN MEMORY FILE SYSTEM IMPLEMENTATION

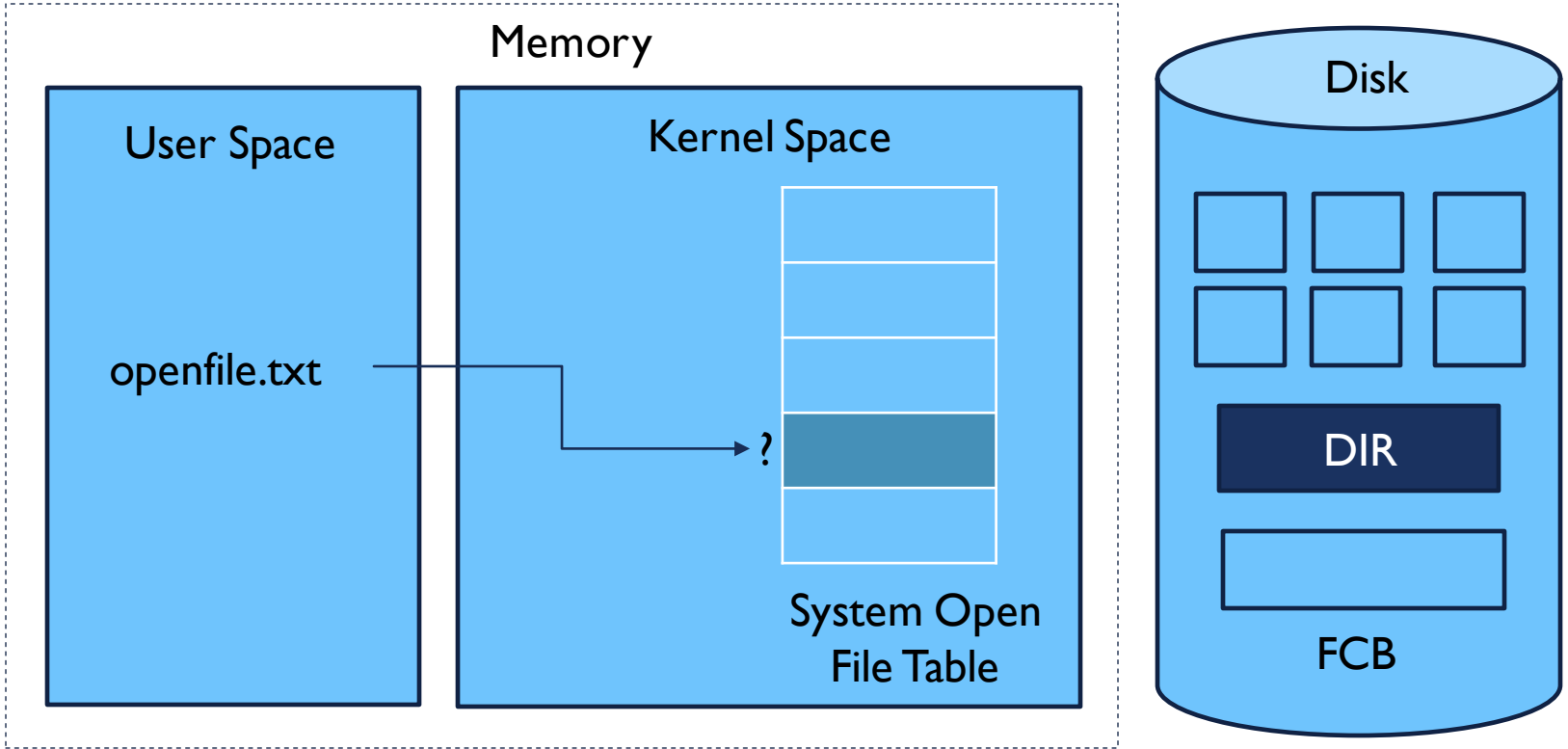
Process of Opening a File



IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

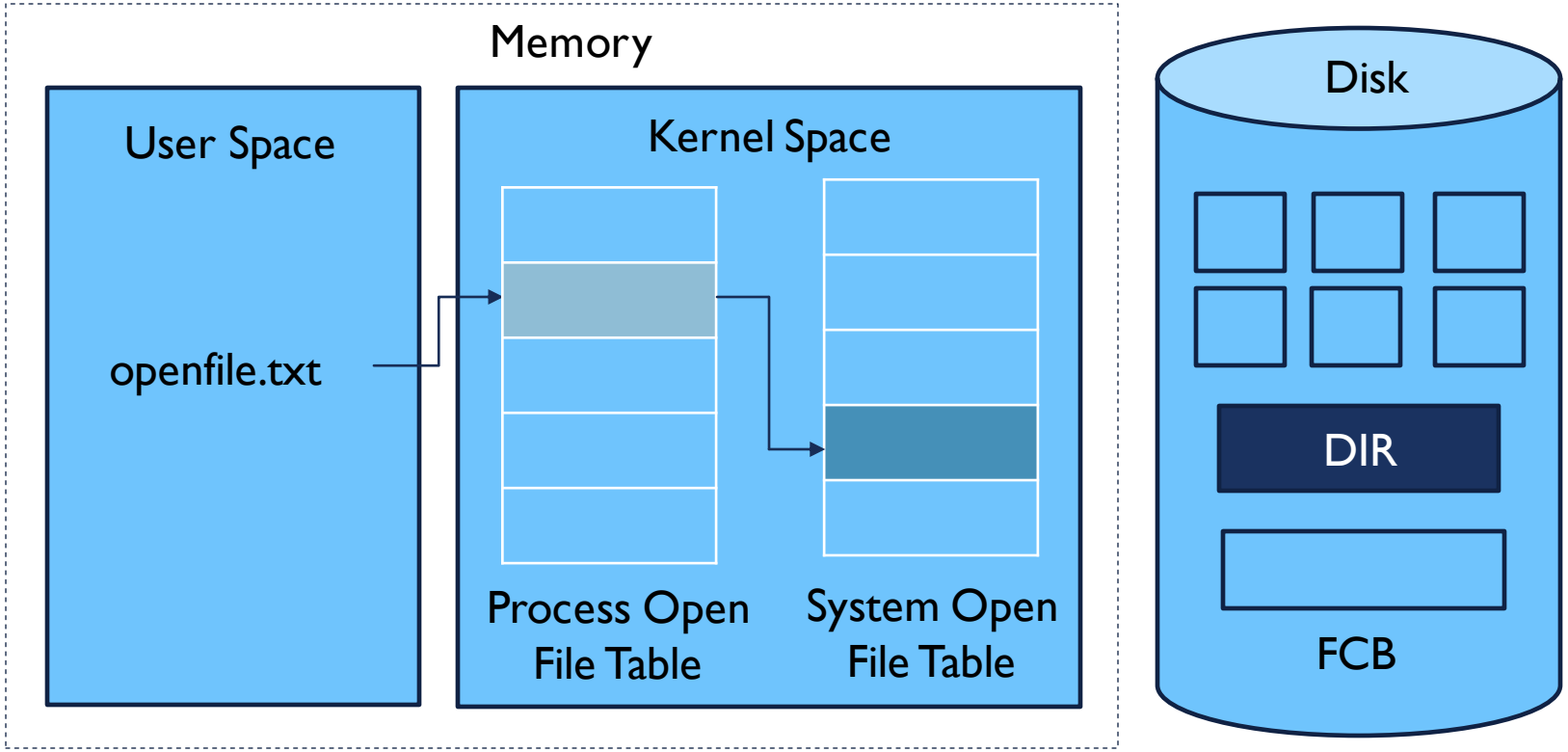
- Search open system file table.



IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

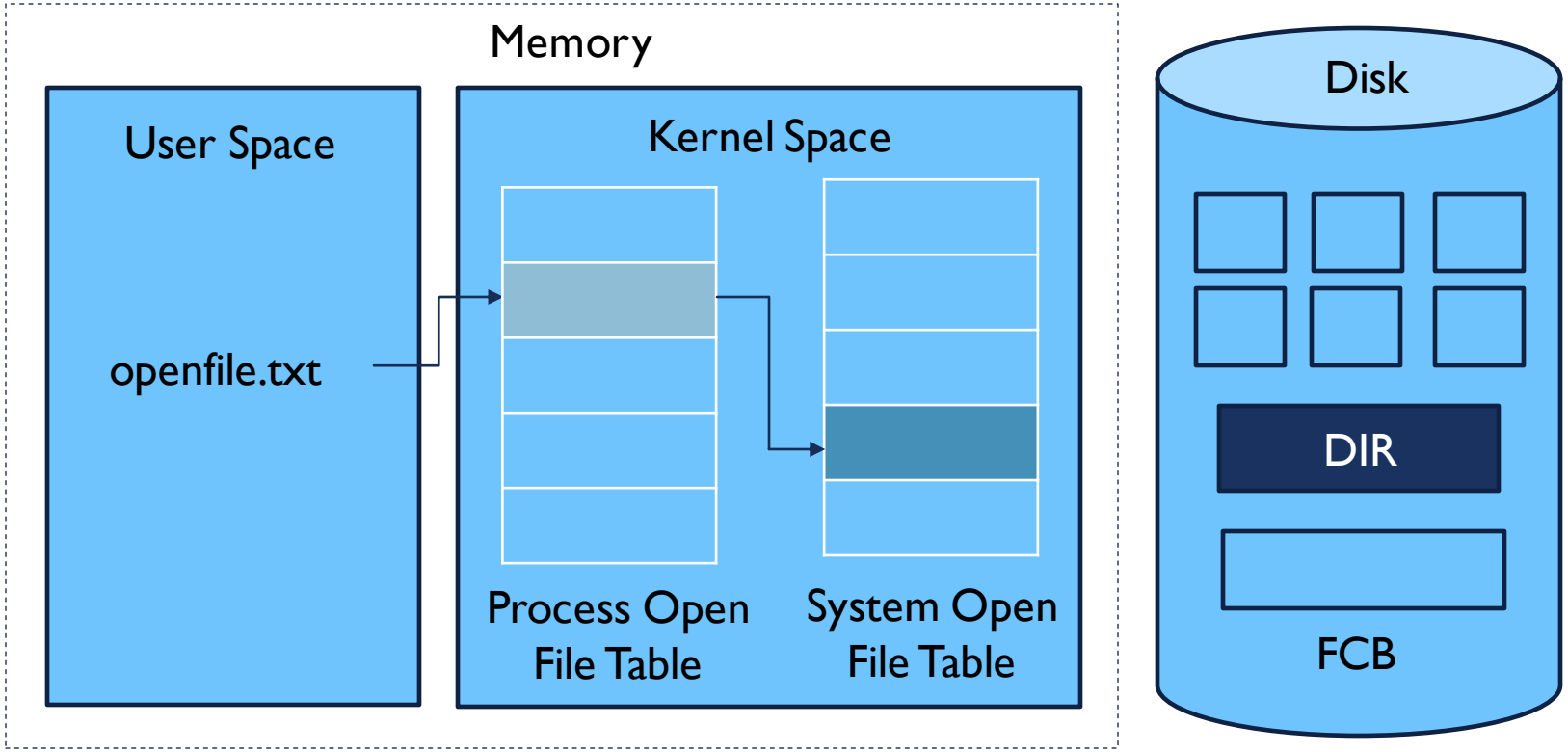
- Search open system file table
- If the file is found in the system table, add to the process table ***a pointer*** to the system table entry.



IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

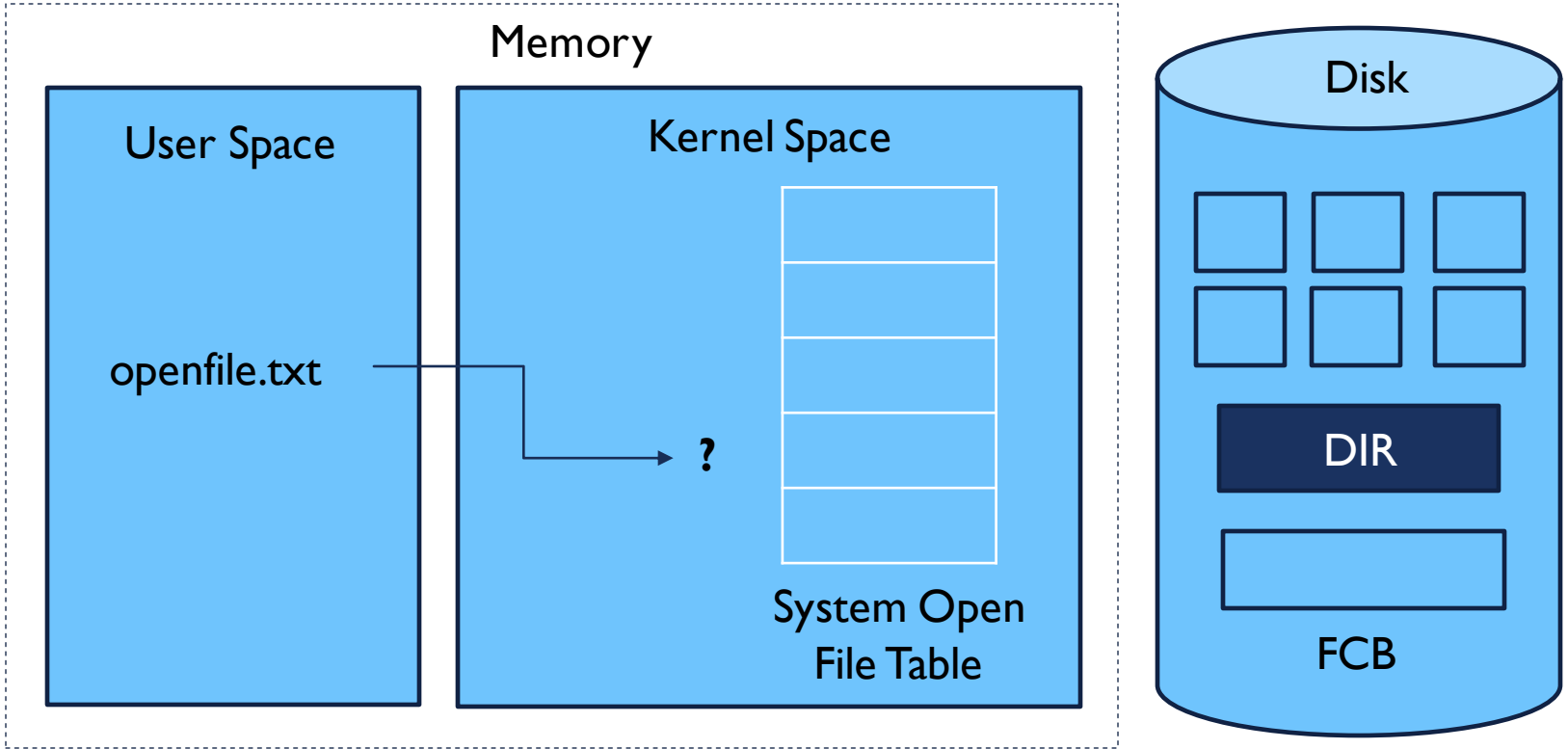
- Search open system file table
- If the file is found in the system table, add to the process table ***a pointer*** to the system table entry.
- System table would contain the File Control Block (FCB) that contains all information about the file.



IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

- If the file is not found in the system open table:

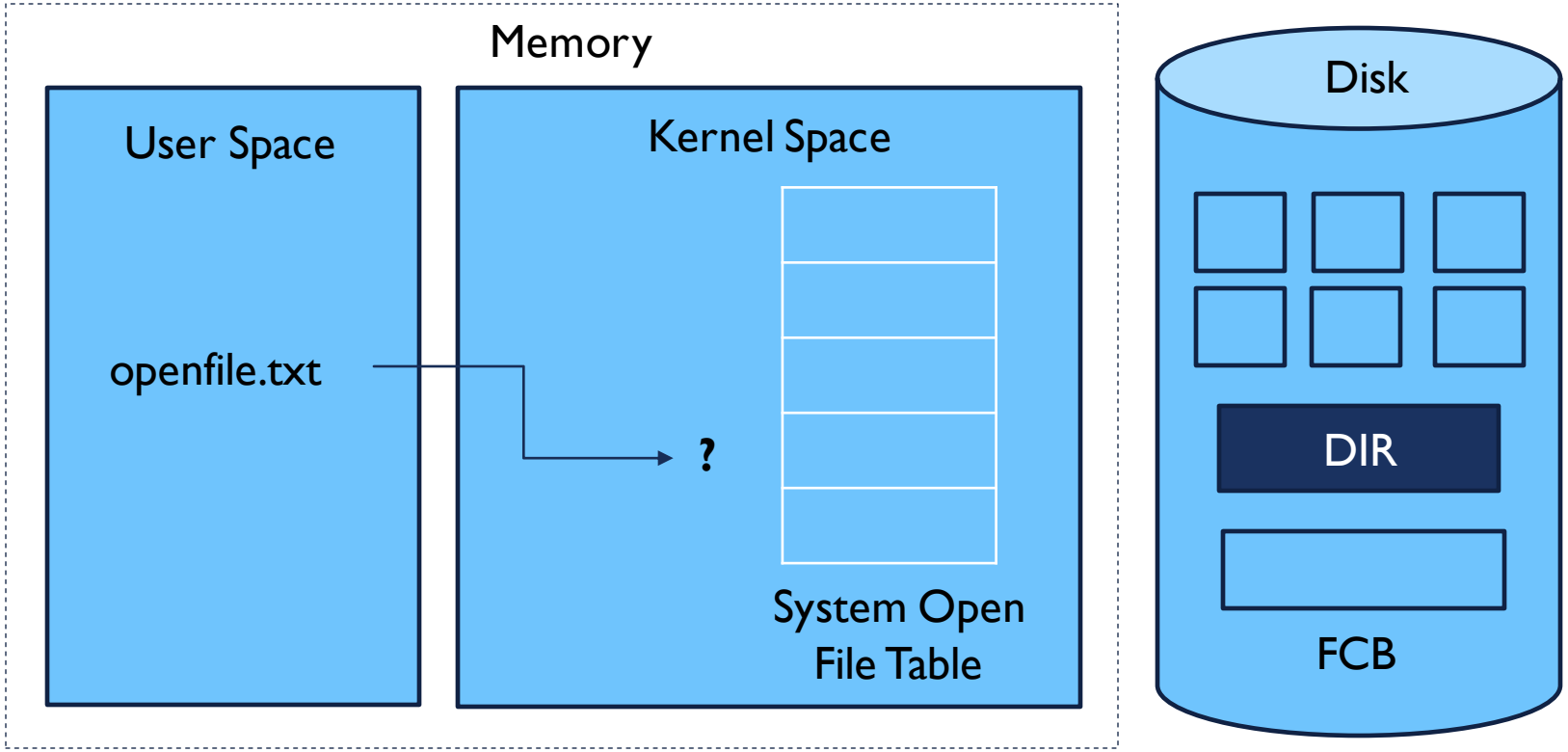


IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

- If the file is not found in the system open table:

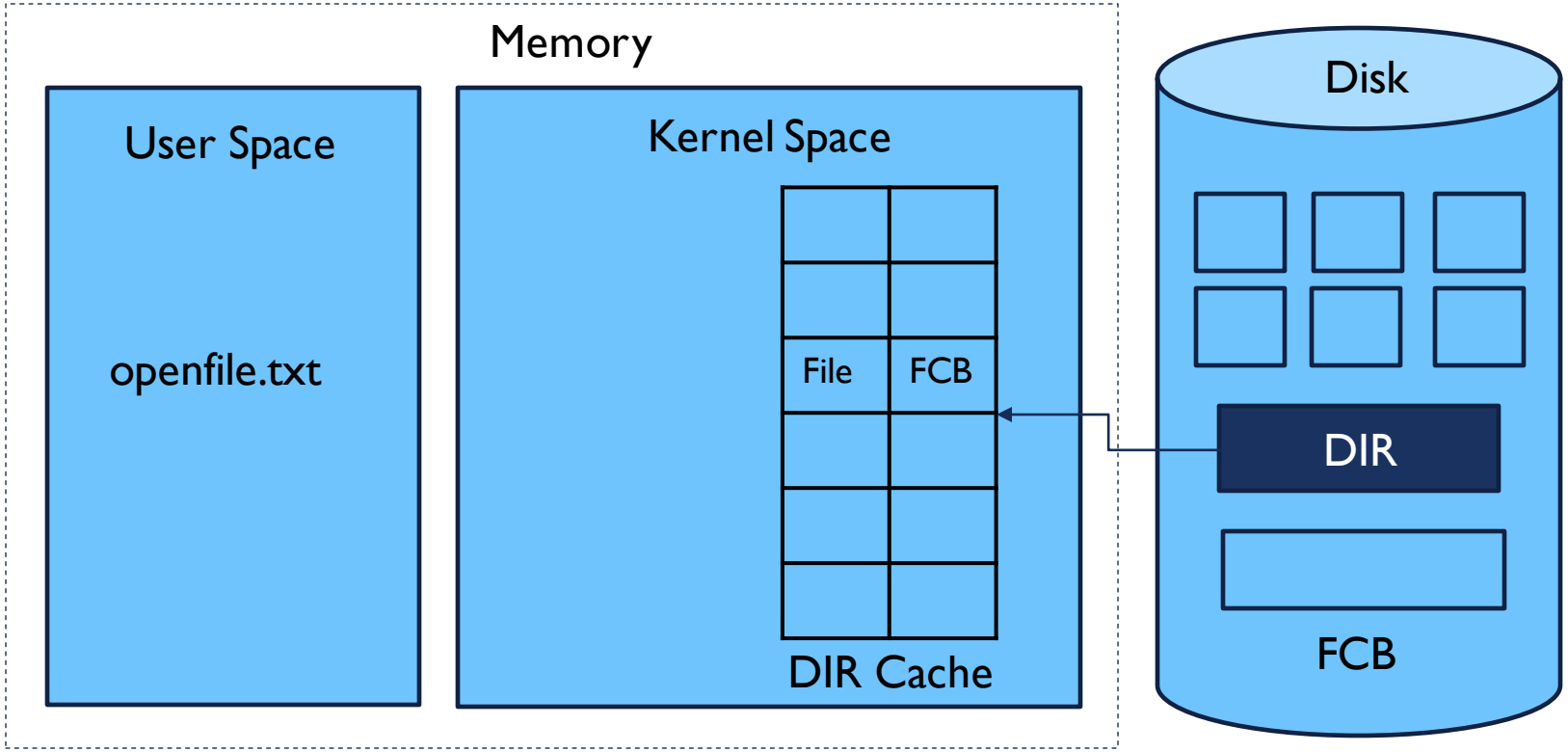
Worksheet Q1: Where and how do we search for it?



IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

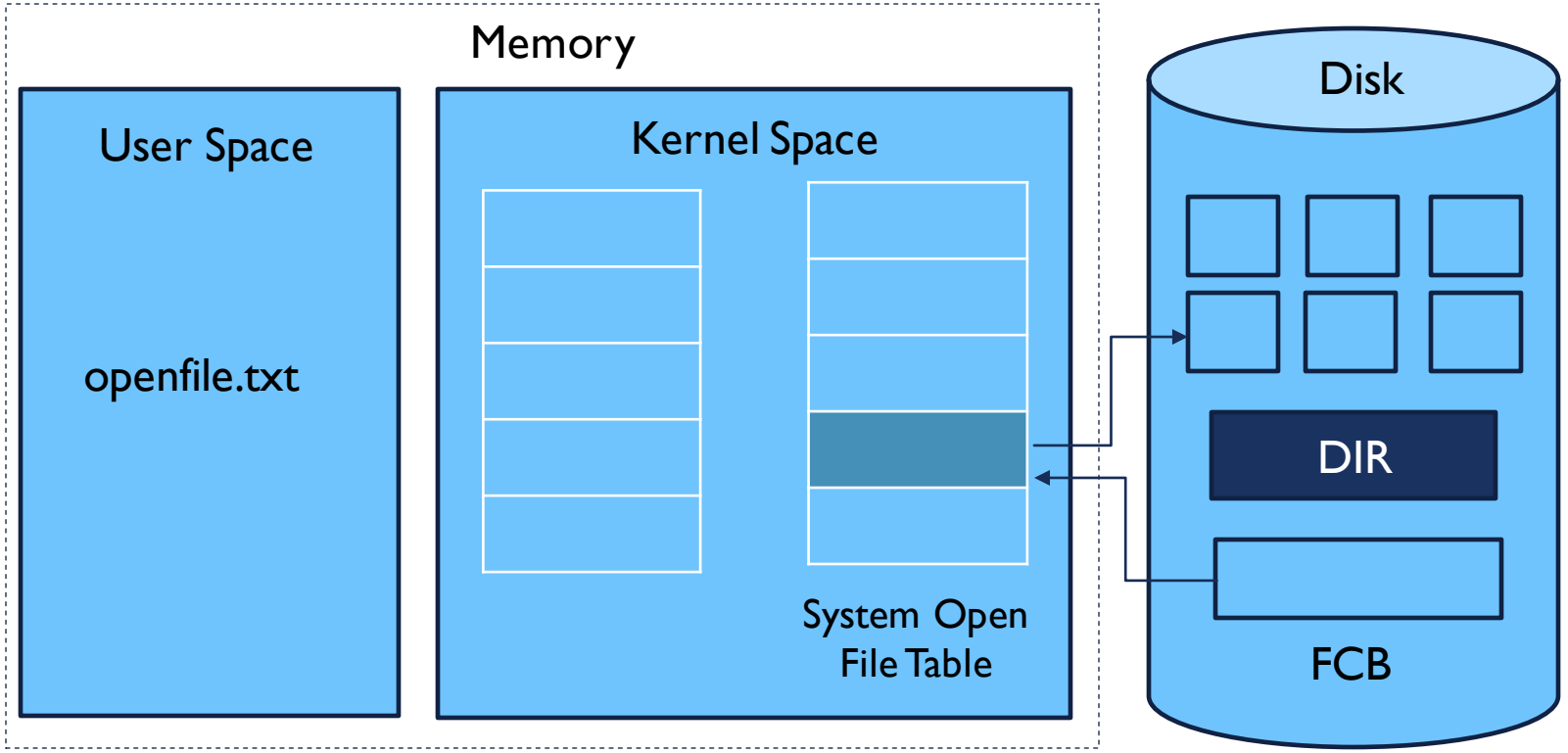
- If the file is not found in the system open table:
- Fetch the directory structure and details from the HD to the directory structure cached in kernel space.



IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

- If the file is not found in the system open table:
- Fetch the directory structure and details from the HD to the directory structure cached in kernel space.
- Set up the FCB so that the kernel space (OS) is aware of it
- Add to system's open files table.



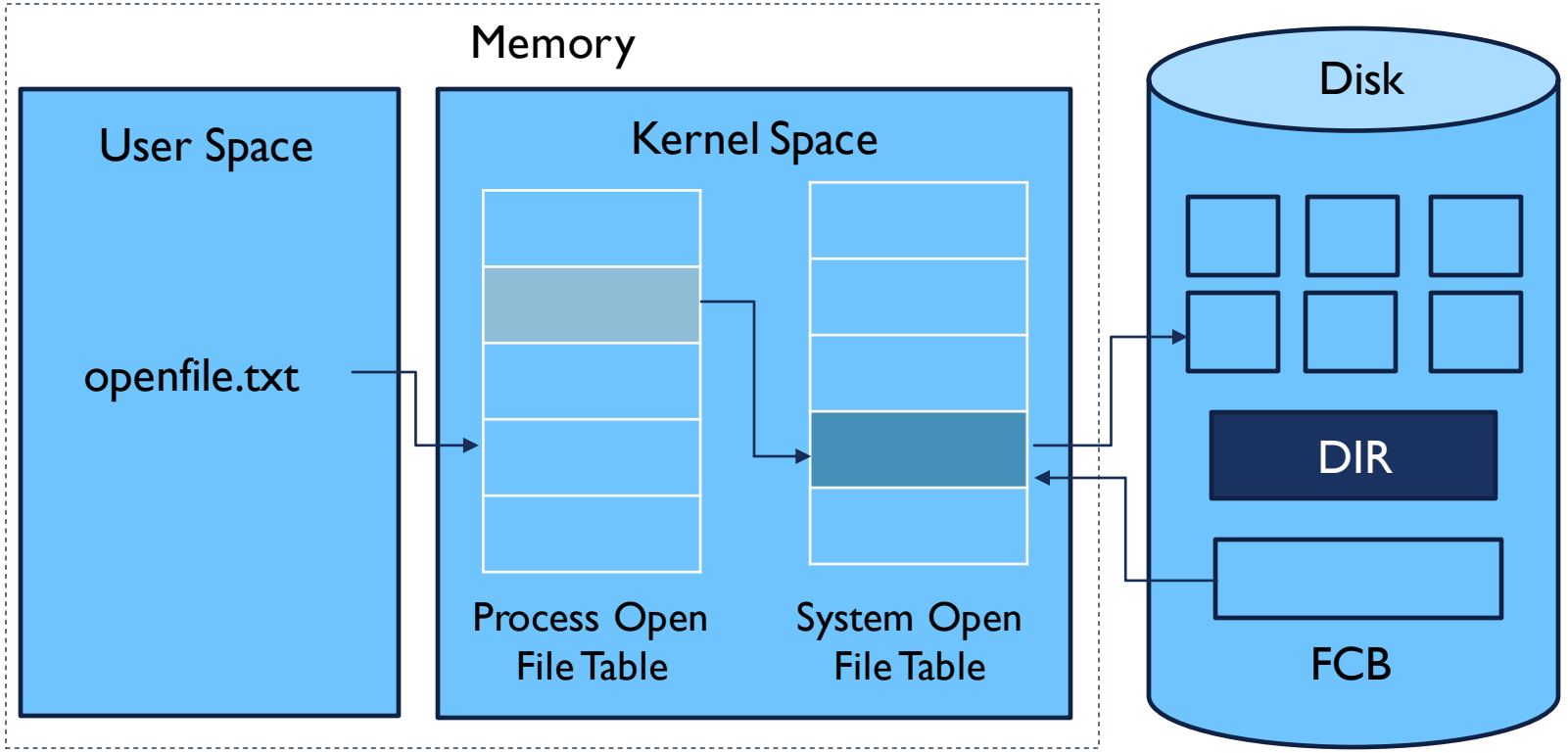
IN MEMORY FILE SYSTEM IMPLEMENTATION

Process of Opening a File

- If the file is not found in the system open table:

- Fetch the directory structure and details from the HD to the directory structure cached in kernel space.
- Set up the FCB so that the kernel space (OS) is aware of it

- Add to system's open files table.
- Set up pointer to system table from process table.

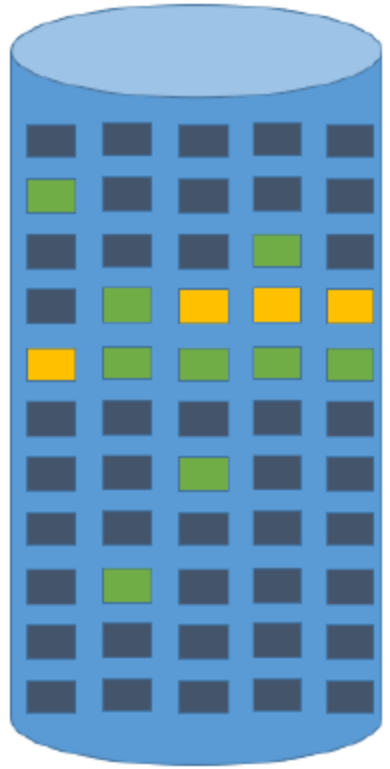


FILE STORAGE ALLOCATION

Q: How should space be allocated?

Q: How should the hard drive's blocks be adjusted / modified, in response to edits to its contents?

For example, if the file that occupies the yellow blocks is appended to … where should the "new" data be placed?



CONTIGUOUS ALLOCATION

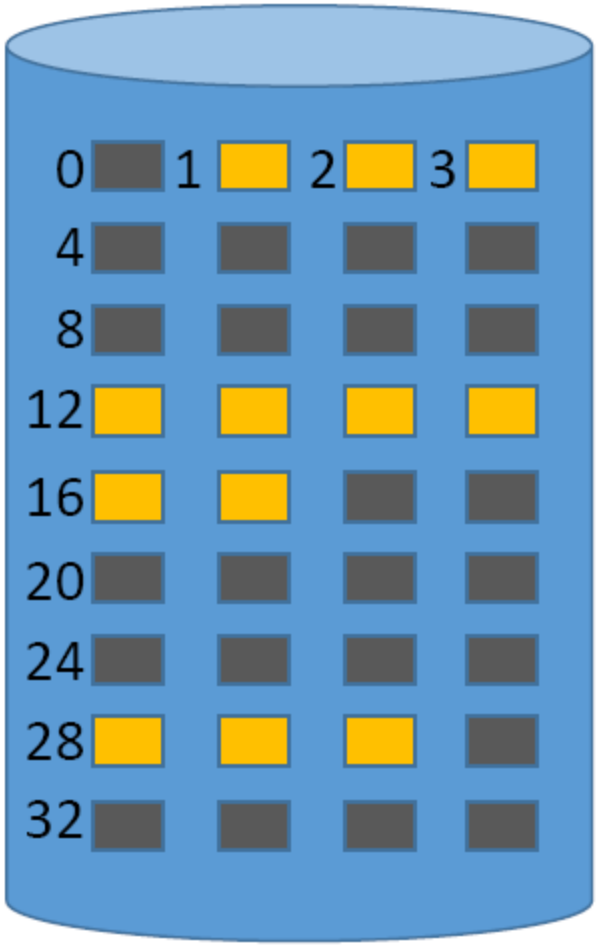
Contiguous Allocation : A file occupies contiguous blocks

Q: Does contiguous block allocation support sequential access?

A B

A: Yes
B: No

directory		
File	**Start**	**Length**
myFile	1	3
aPic	12	6
song	28	3



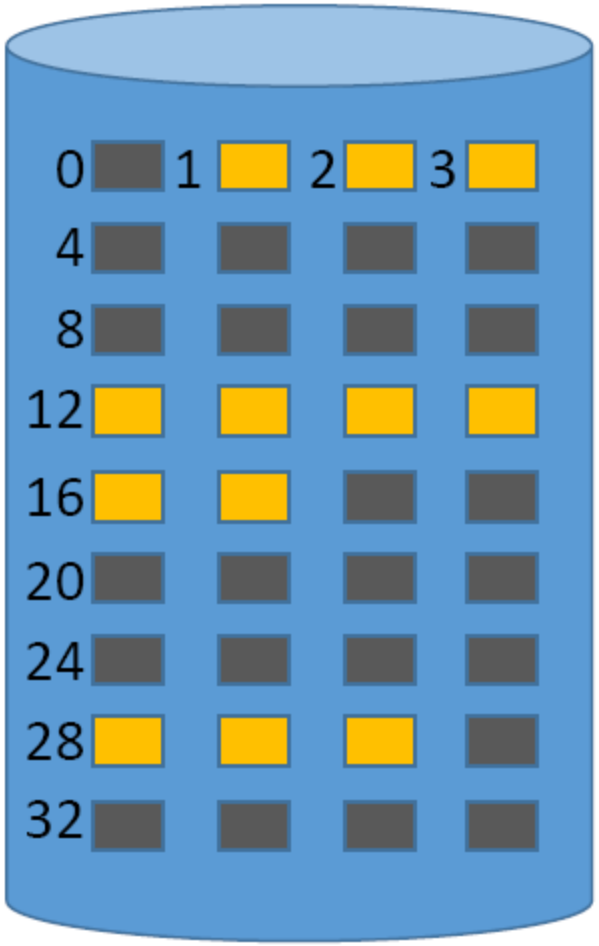
CONTIGUOUS ALLOCATION

Contiguous Allocation : A file occupies contiguous blocks

Q: Does contiguous block allocation support sequential access?

Q: Does contiguous block allocation support direct access?

directory		
File	**Start**	**Length**
myFile	1	3
aPic	12	6
song	28	3



A B

A: Yes
B: No

CONTIGUOUS ALLOCATION

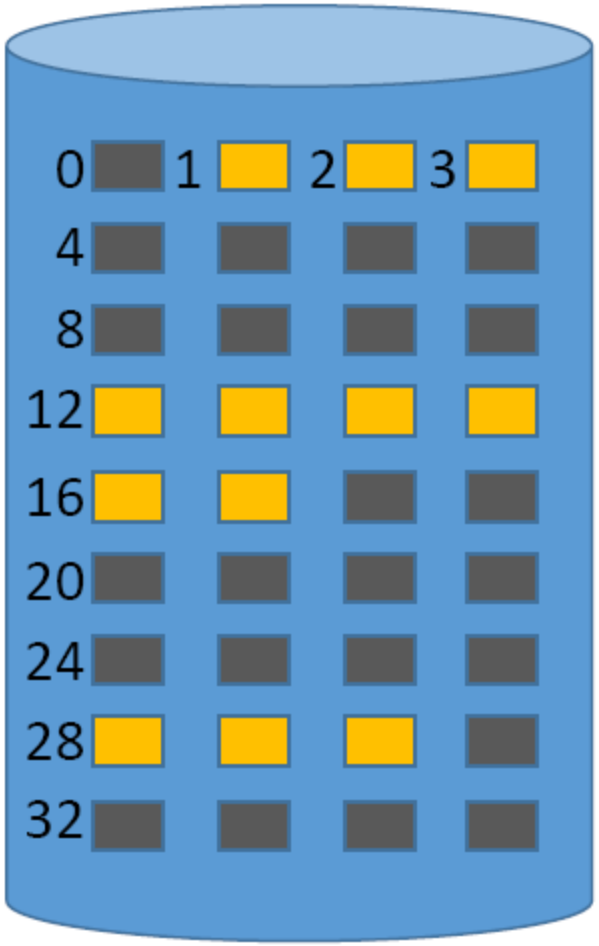
Contiguous Allocation : A file occupies contiguous blocks

Q: Does contiguous block allocation support sequential access?

Q: Does contiguous block allocation support direct access?

directory

File	Start	Length
myFile	1	3
aPic	12	6
song	28	3

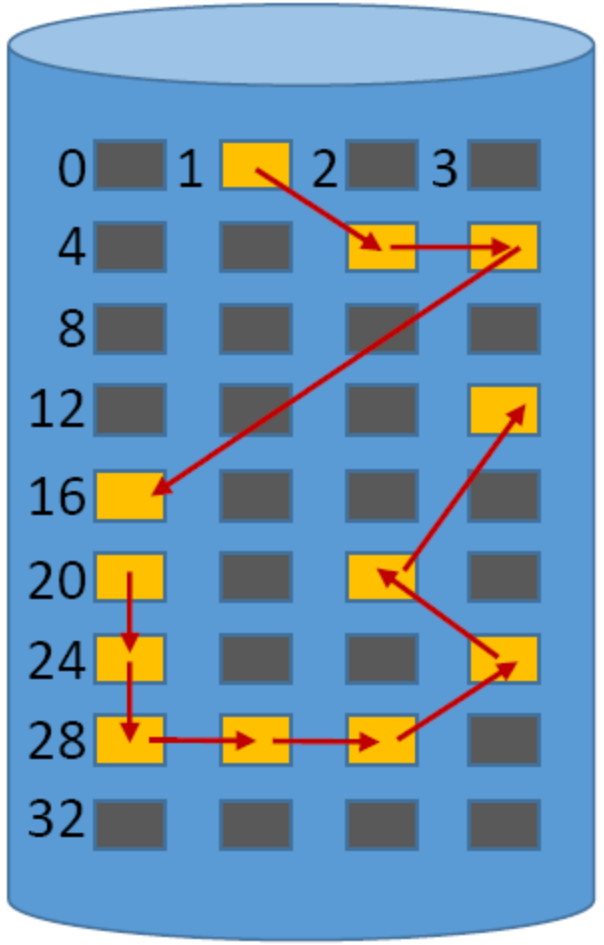


Q: Can identify a particular challenge or drawback for contiguous allocation?

Fragmentation

LINKED ALLOCATION

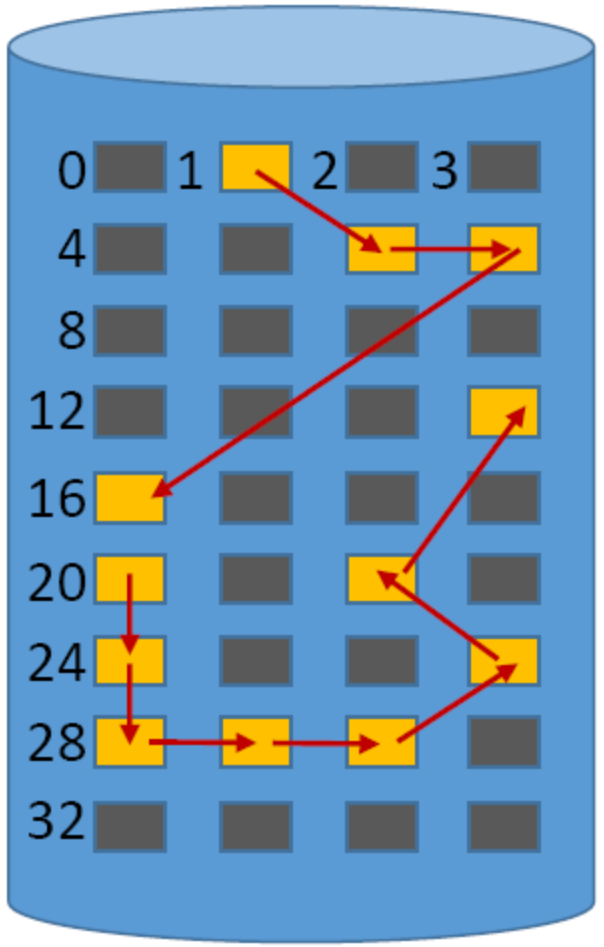
Linked Allocation : A file is a linked list of blocks



LINKED ALLOCATION

Linked Allocation : A file is a linked list of blocks

directory		
File	**Start**	**End**
myFile	1	16
aPic	20	15

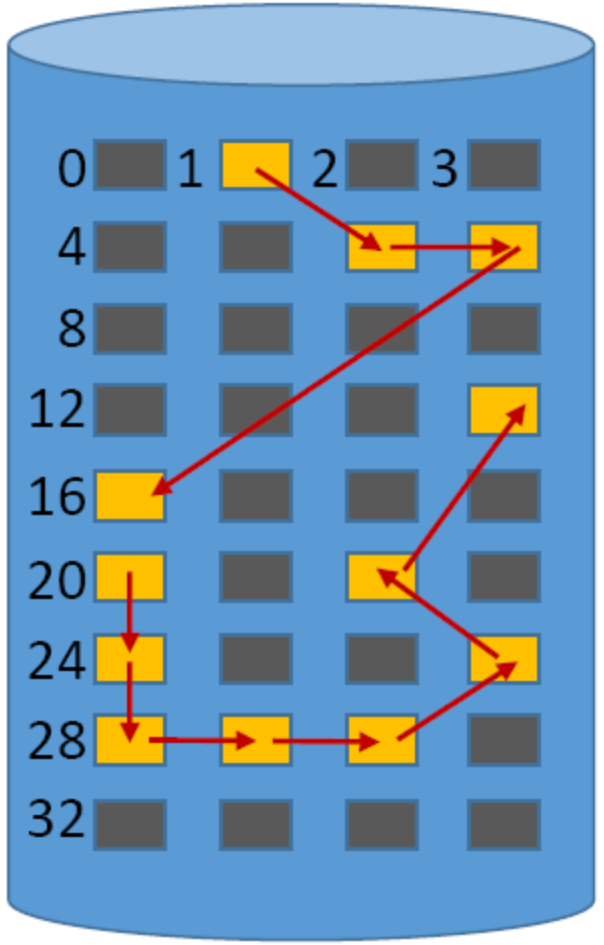


LINKED ALLOCATION

Linked Allocation : A file is a linked list of blocks

Worksheet Q1:What would be the advantages/disadvantages?

directory		
File	**Start**	**End**
myFile	1	16
aPic	20	15

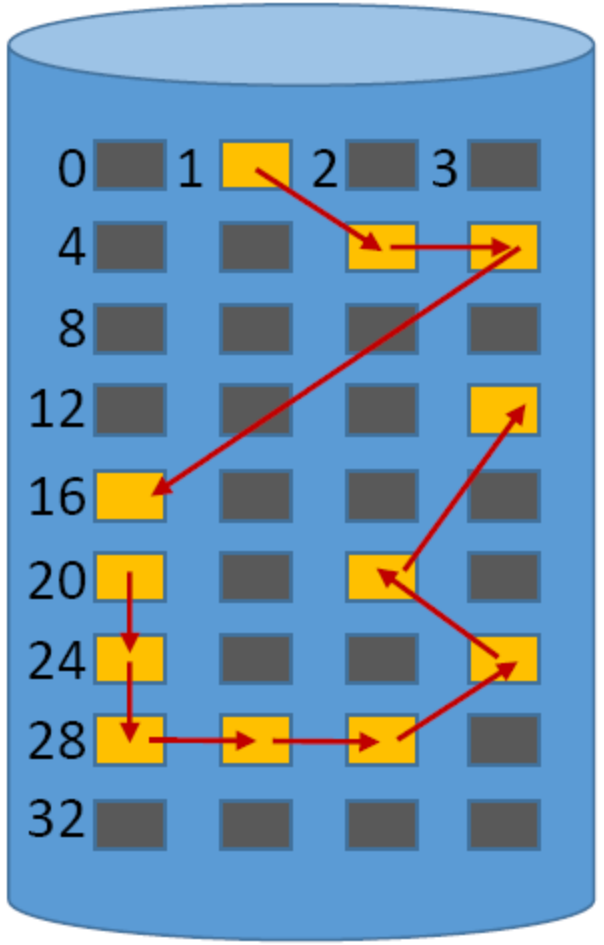


LINKED ALLOCATION

Linked Allocation : A file is a linked list of blocks

If a block is of size *x*, then how much data can you store in *x*?

directory		
File	**Start**	**End**
myFile	1	16
aPic	20	15

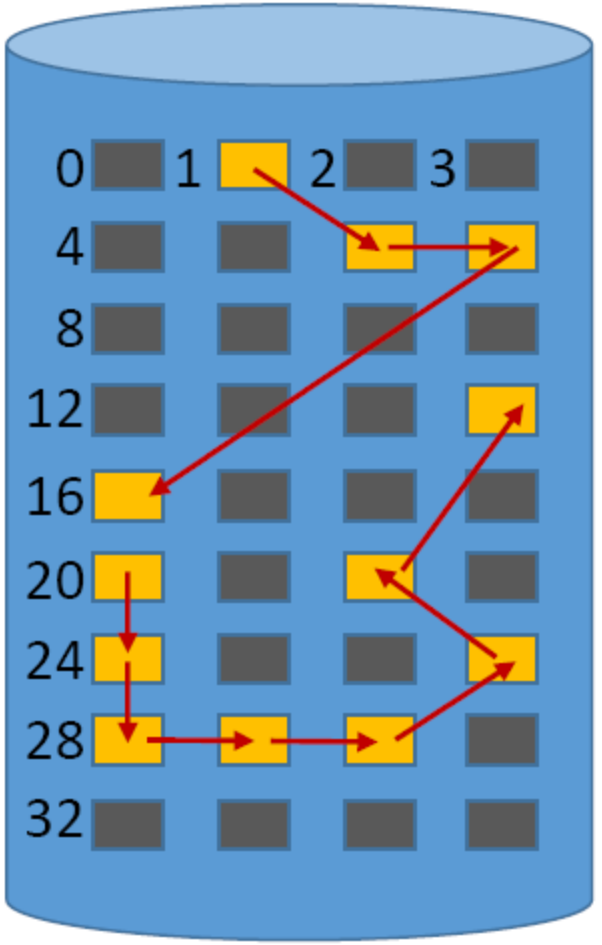


LINKED ALLOCATION

Linked Allocation : A file is a linked list of blocks

Each block's size for storing data is reduced by the amount of bytes needed to store an address

directory		
File	**Start**	**End**
myFile	1	16
aPic	20	15



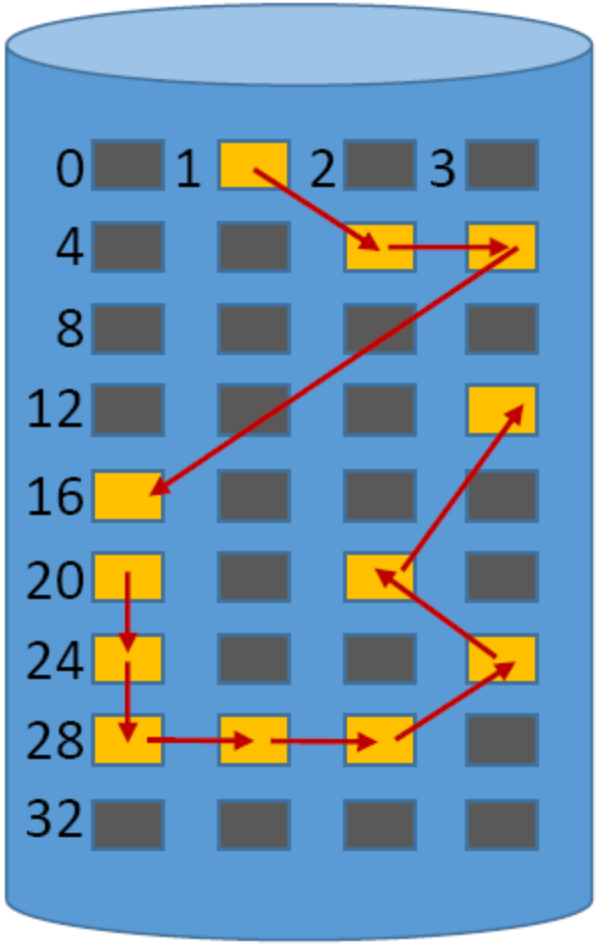
LINKED ALLOCATION

Linked Allocation : A file is a linked list of blocks

Each block's size for storing data is reduced by the amount of bytes needed to store an address

This solves the fragmentation problem contiguous allocation had.

directory		
File	**Start**	**End**
myFile	1	16
aPic	20	15



LINKED ALLOCATION

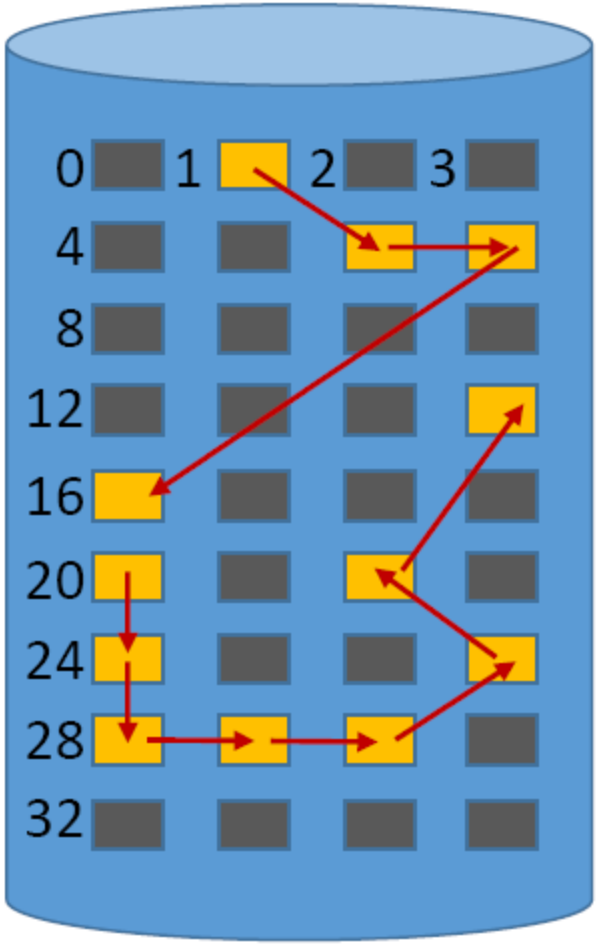
Linked Allocation : A file is a linked list of blocks

Each block's size for storing data is reduced by the amount of bytes needed to store an address

This solves the fragmentation problem contiguous allocation had.

directory		
File	**Start**	**End**
myFile	1	16
aPic	20	15

Q: What are the drawbacks of linked block allocation?



LINKED ALLOCATION

Linked Allocation : A file is a linked list of blocks

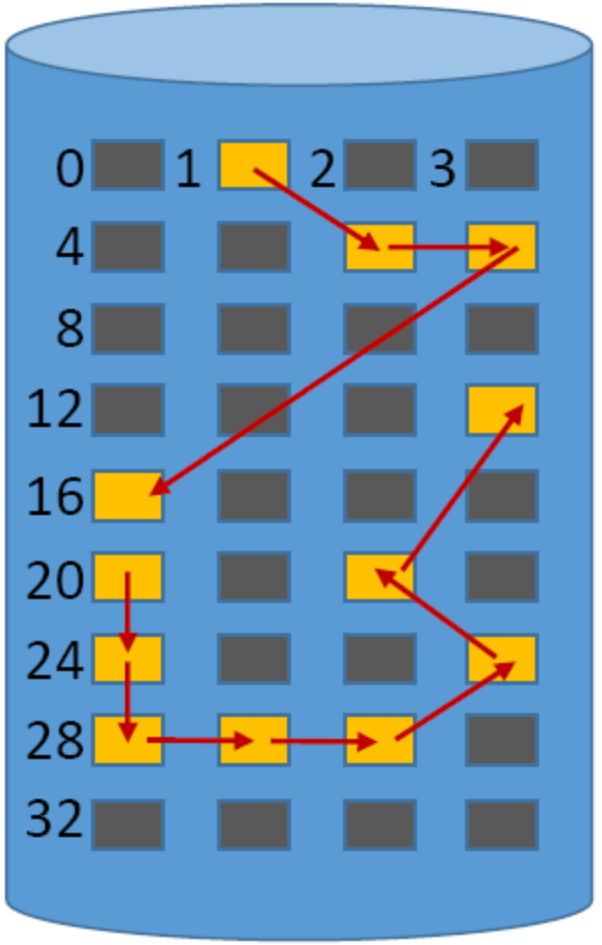
Each block's size for storing data is reduced by the amount of bytes needed to store an address

This solves the fragmentation problem contiguous allocation had.

directory		
File	**Start**	**End**
myFile	1	16
aPic	20	15

Disadvantages:

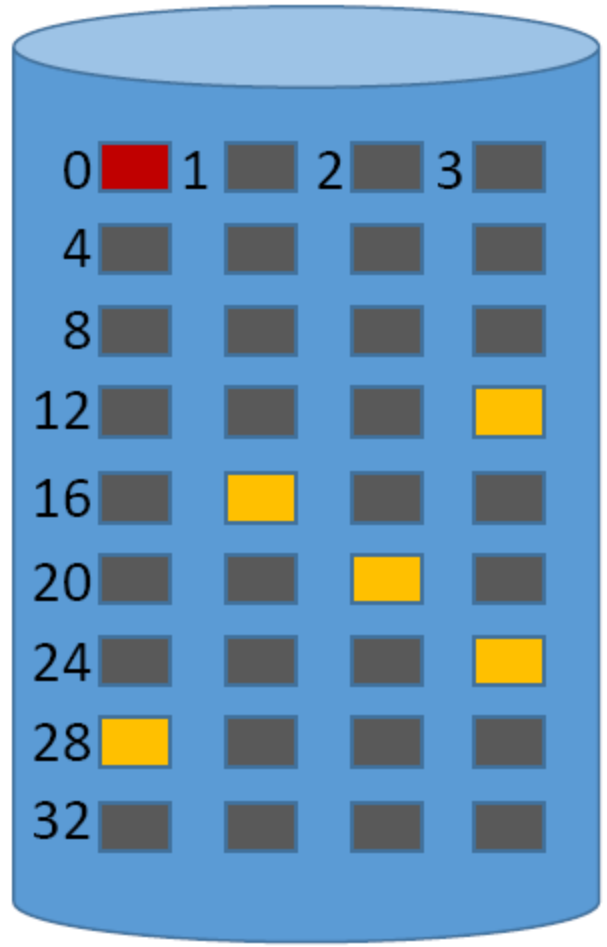
- No support for direct access
- Block size effectively reduced
- Slower disk performance
- More difficult recovering from error; lost links might lead to lost files.



FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

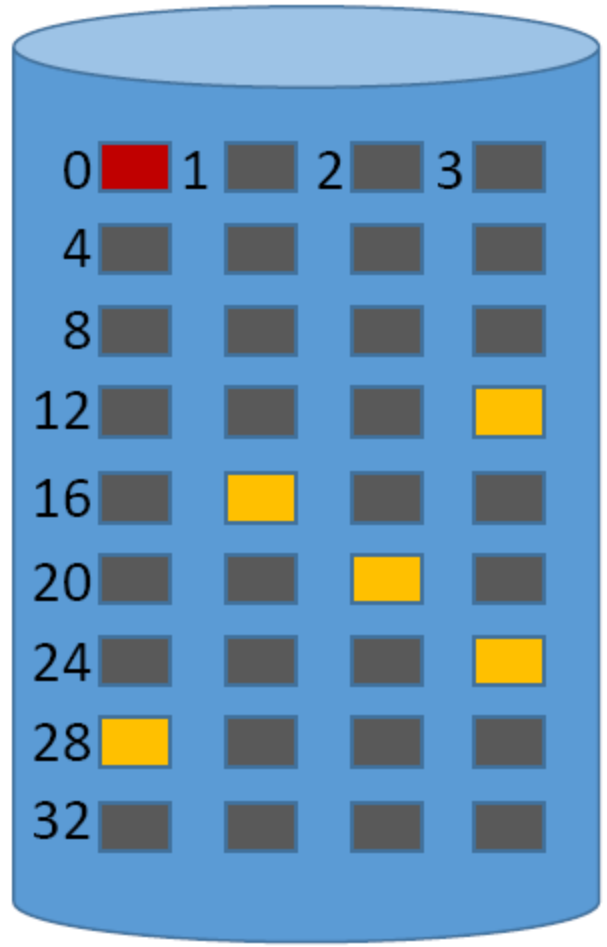


FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

Q: What should we put into the FAT?



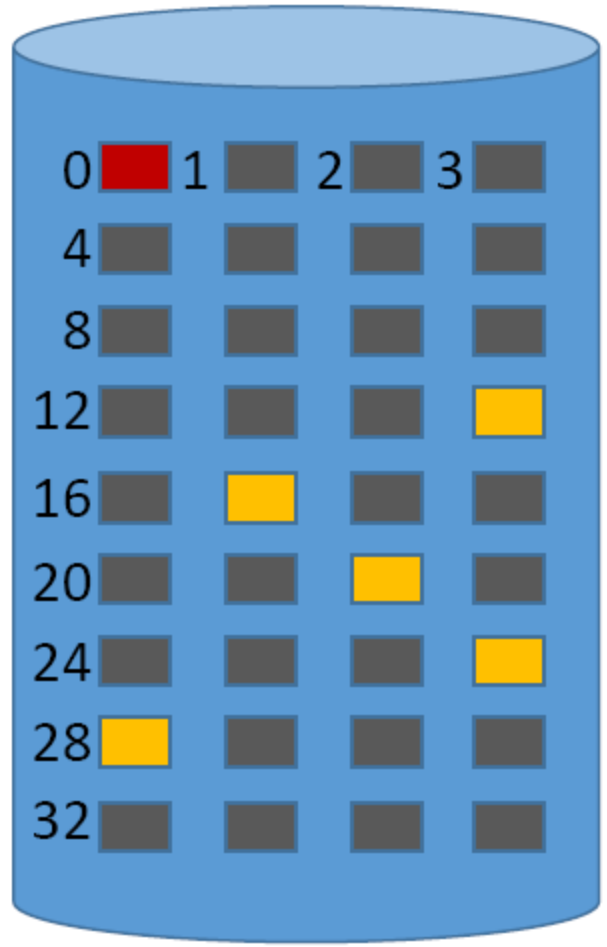
FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

Directory entry

aFile	other	17



FILE ALLOCATION TABLE

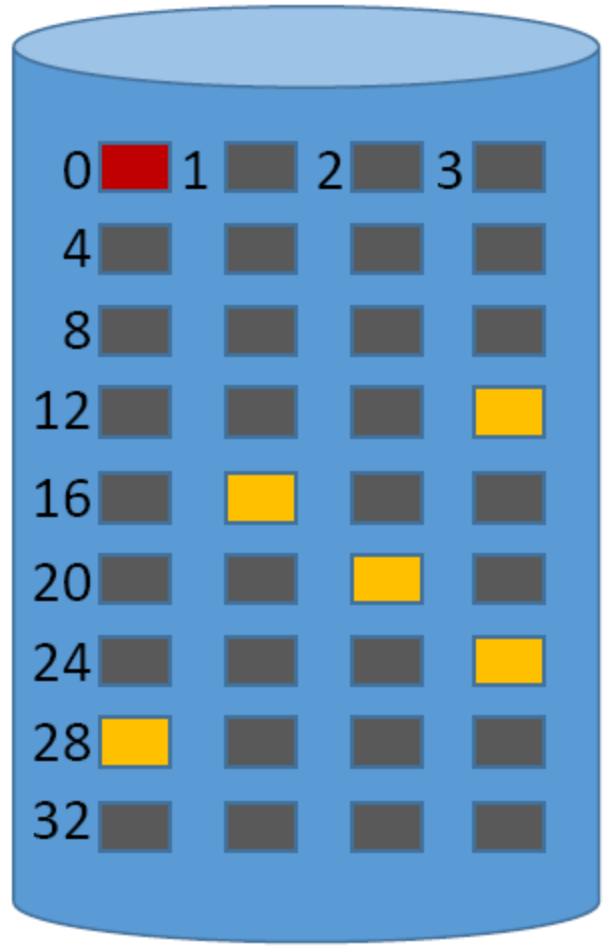
File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

Directory entry

aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies



FILE ALLOCATION TABLE

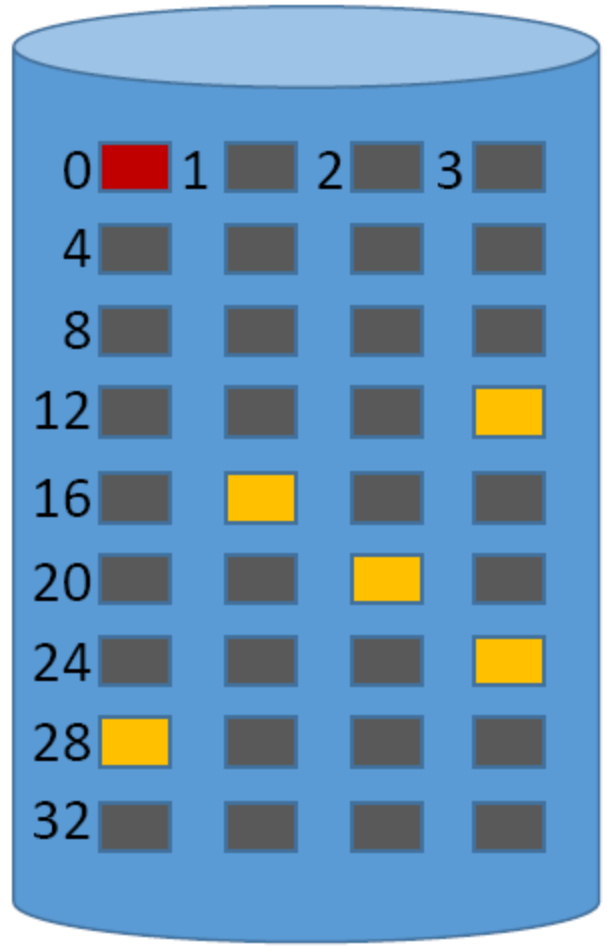
File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

Directory entry

aFile	other	17

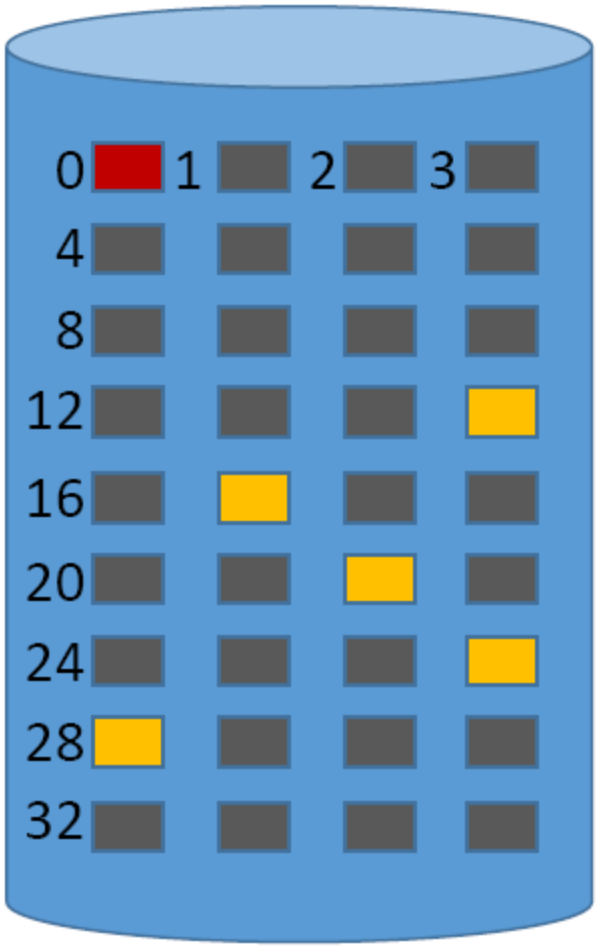
The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies



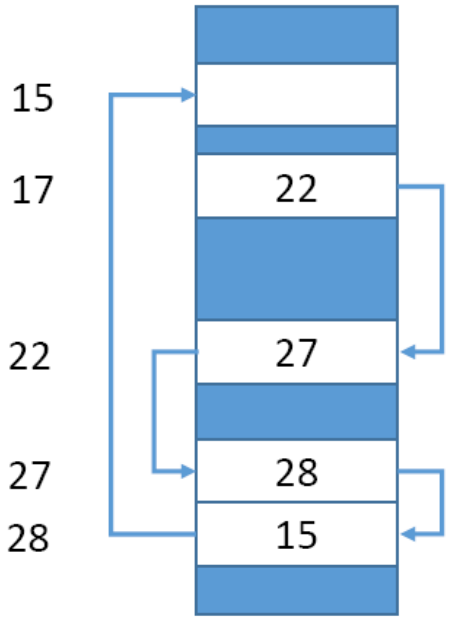
FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks



File Allocation Table



Directory entry

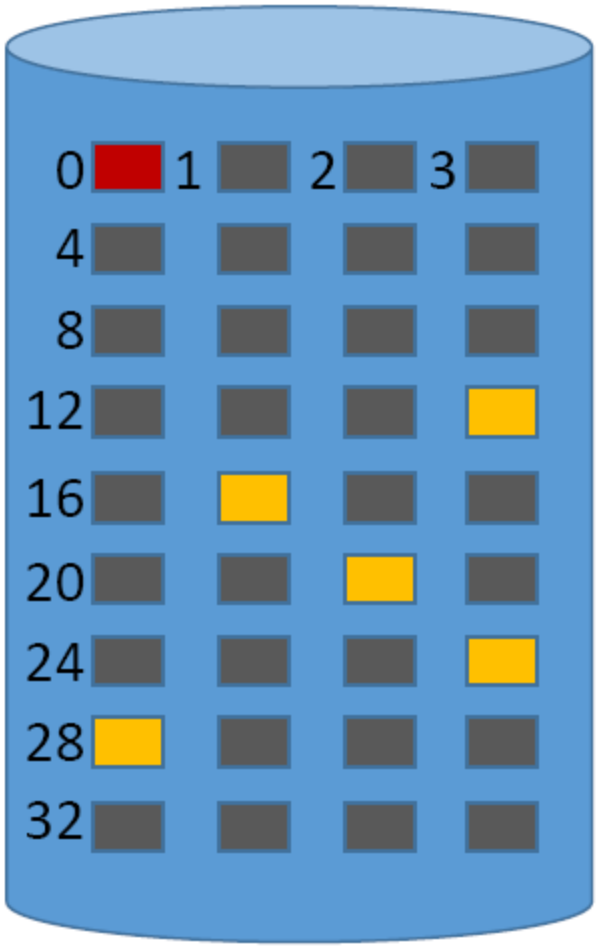
aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies

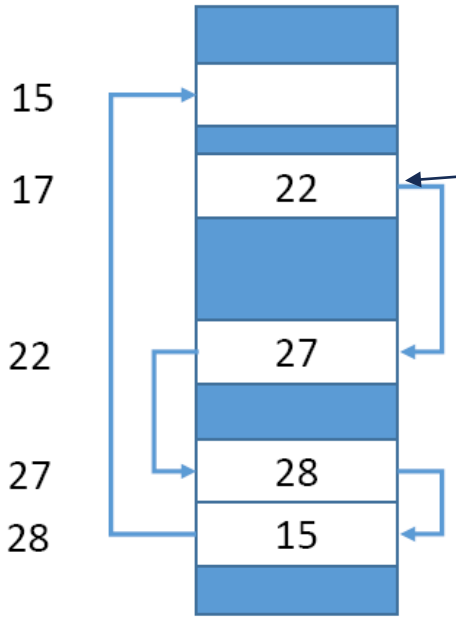
FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks



File Allocation Table



Directory entry

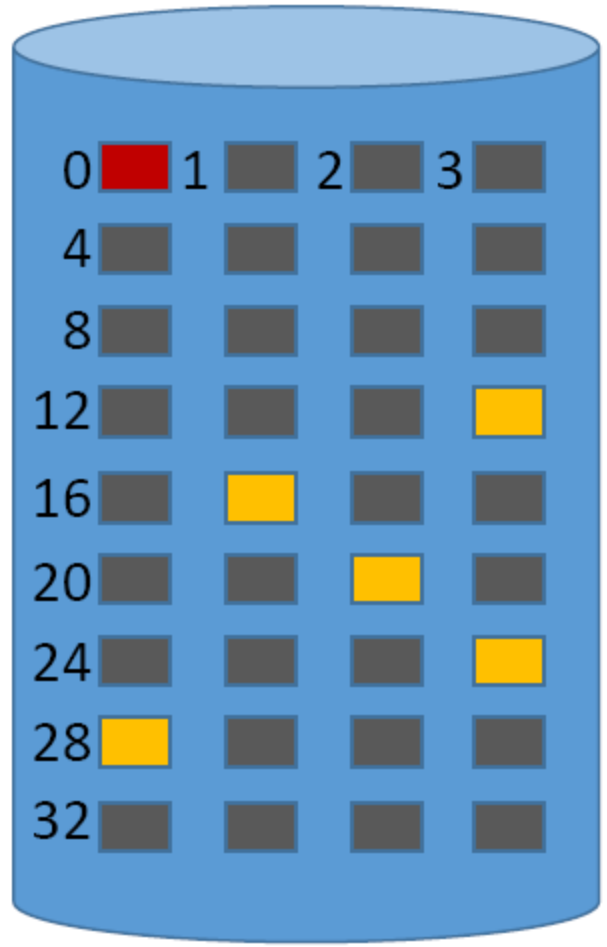
aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies

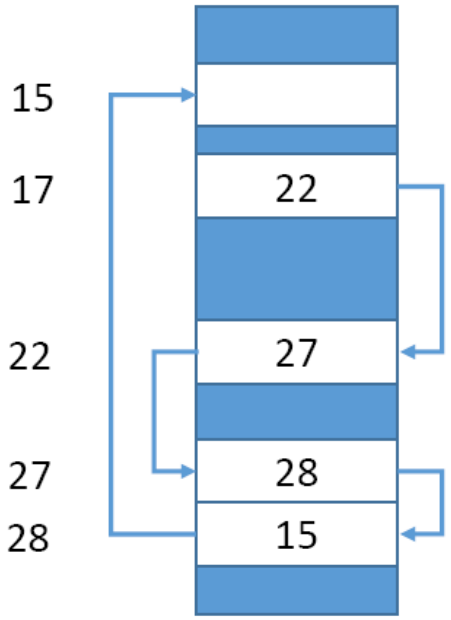
FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks



File Allocation Table



- The "chaining" is only done in the table.
- Allows for faster traverse.

Directory entry

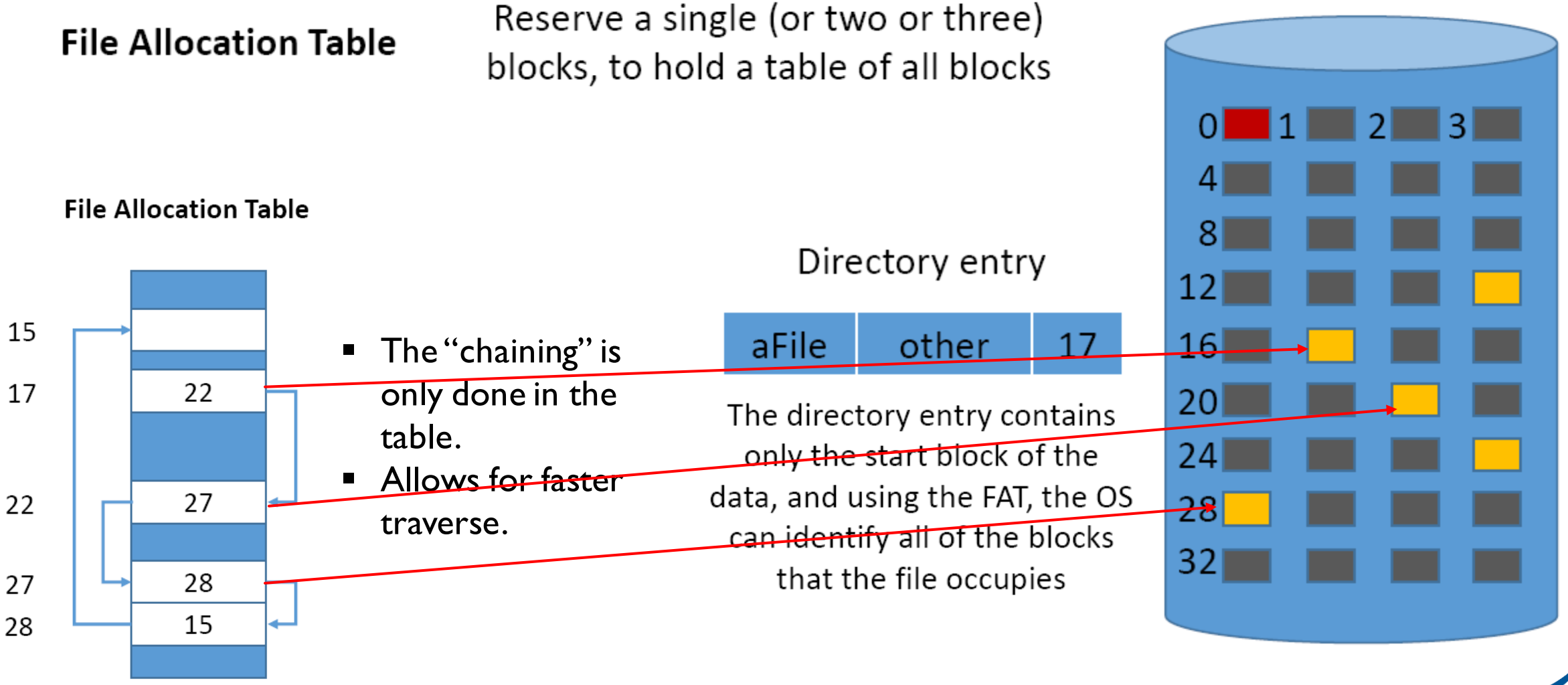
aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies

FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

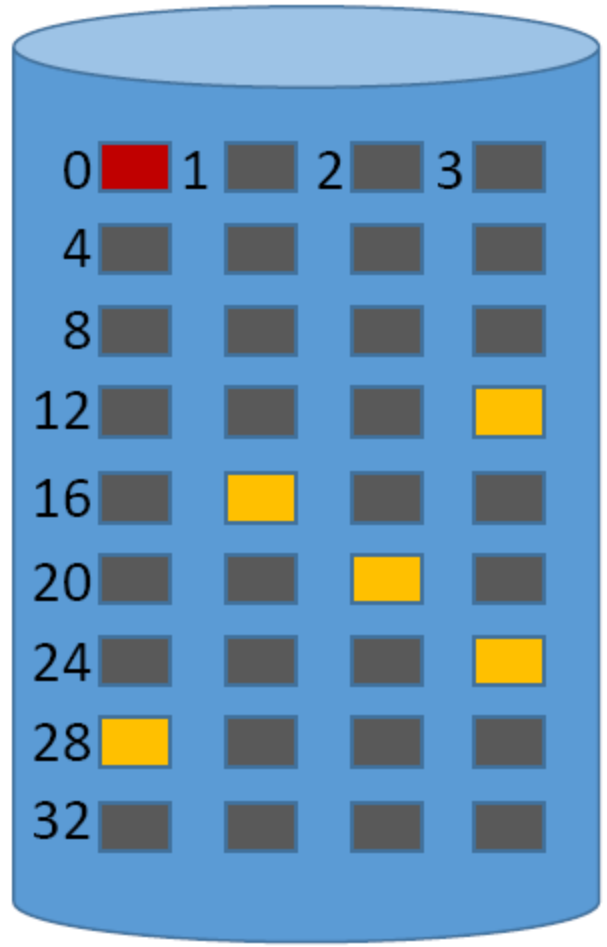


- The "chaining" is only done in the table.
- Allows for faster traverse.

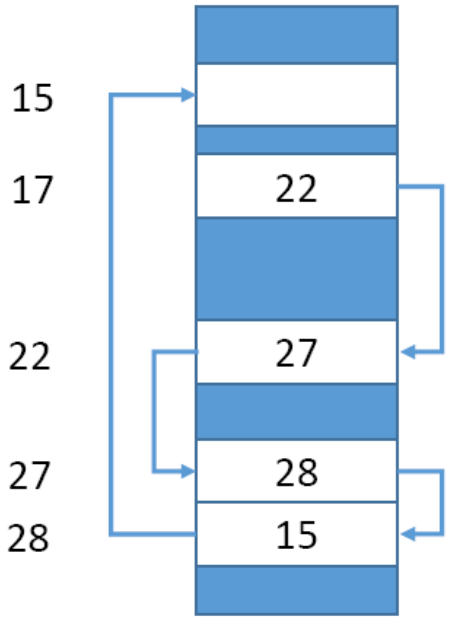
FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks



File Allocation Table



- The "chaining" is only done in the table.
- Allows for faster traverse.

Directory entry

aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies