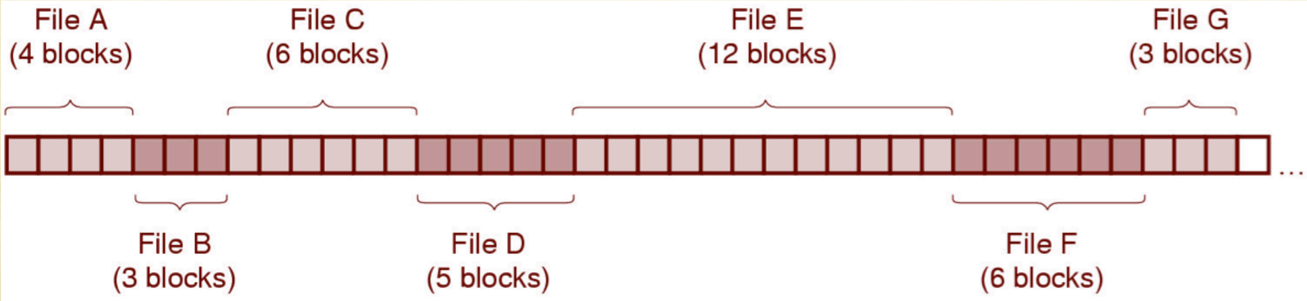
File allocation – contiguous

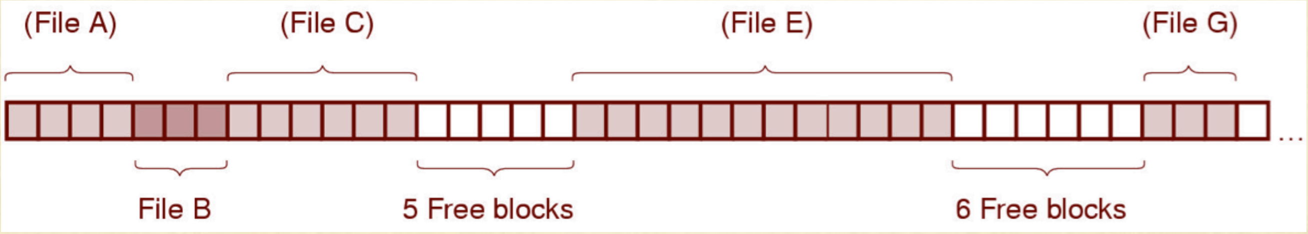
- File blocks stored *contiguously*
- *Pros*:
  - Simple
  - Read is very *fast*



Contiguous allocation of disk space for 7 files

File allocation – contiguous

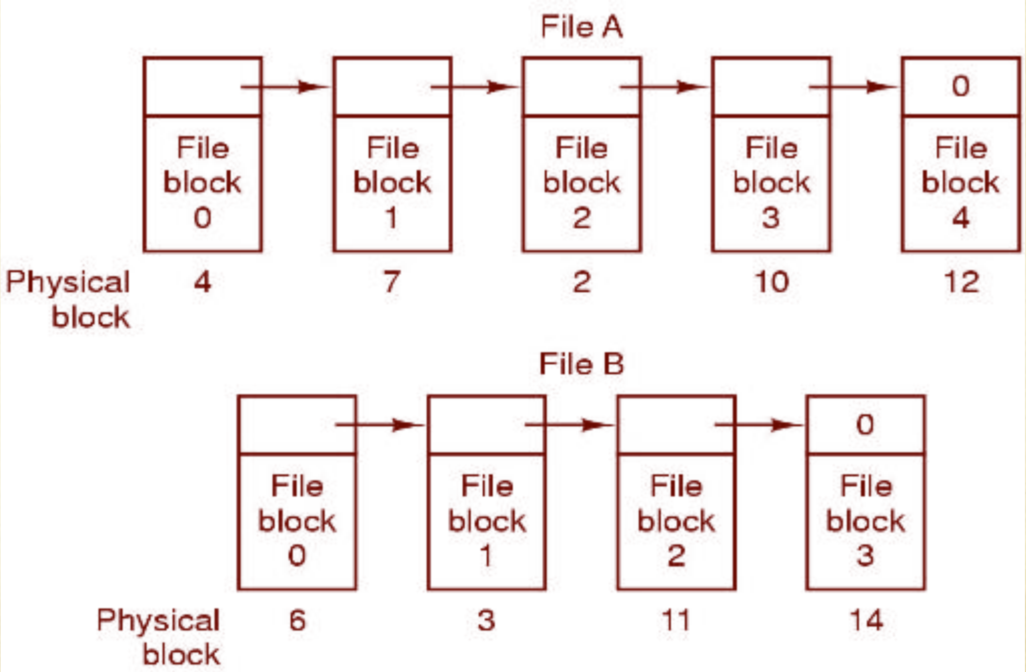
- *Cons*:
  - Results in *external fragmentation*
    - Compaction or de-fragmentation very *expensive*!
  - Max file *size* must be known



State of disk after files D and F have been removed

File allocation – linked list

- File blocks form linked list
- Info about next block stored *within disk block* itself

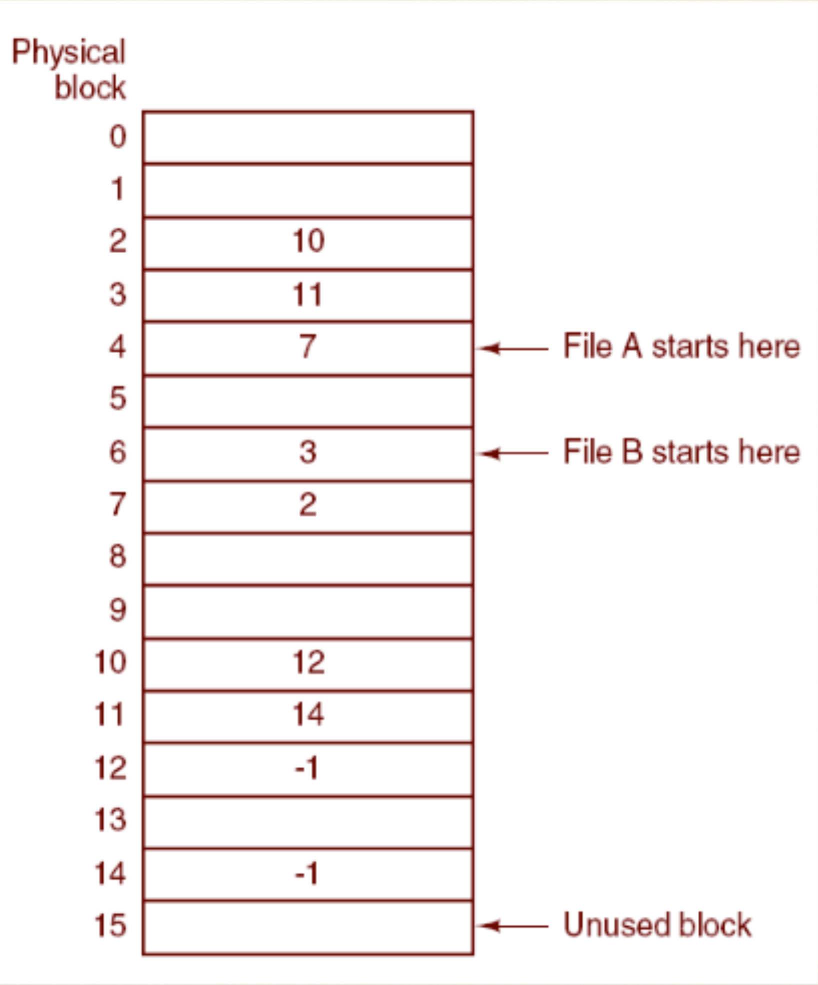


File allocation – linked list

- *Pros*:
  - No *external* fragmentation
  - File size can *dynamically* change
- *Cons*:
  - Could have *internal fragmentation* in last block
  - *Entire* disk block not used for file content
  - To locate random block in file, several *disk* accesses needed!

File allocation – linked list

- Linked list info stored as *table* in *main memory*
  - File allocation table (*FAT*)
  - Disk blocks *entirely* used for file content
  - External pointers to *first* & possibly *last* blocks
  - Each block points to *next* block (or special EOF value)



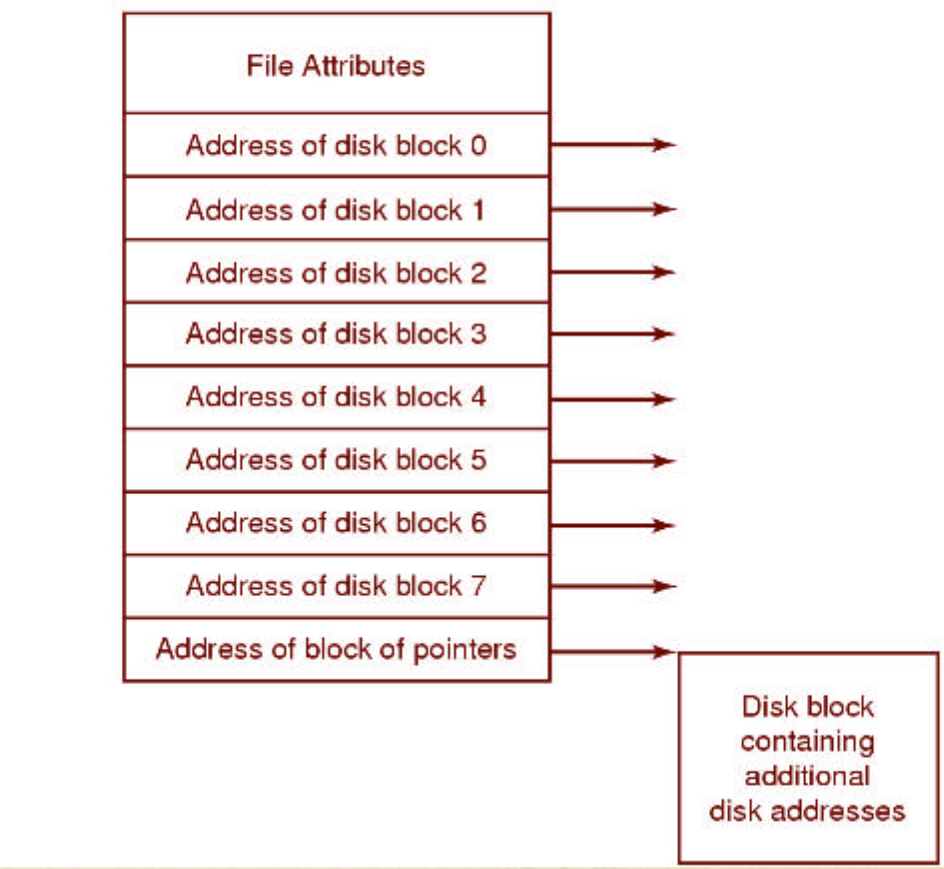
Physical block		
0		
1		
2	10	
3	11	
4	7	← File A starts here
5		
6	3	← File B starts here
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		← Unused block

File allocation – linked list

- *Pro*:
  - To locate random block within file, only *memory* accesses needed
- *Con*:
  - *Entire* table must be in main memory!

File allocation – i-node

- *Attributes* & disk block *addresses* of file stored in data structure called *i-node*
- When file *opened*, its i-node brought into *main memory*
    - I.e., only i-nodes of *open* files kept in main memory



File Attributes

Address of disk block 0

Address of disk block 1

Address of disk block 2

Address of disk block 3

Address of disk block 4

Address of disk block 5

Address of disk block 6

Address of disk block 7

Address of block of pointers

Disk block containing additional disk addresses