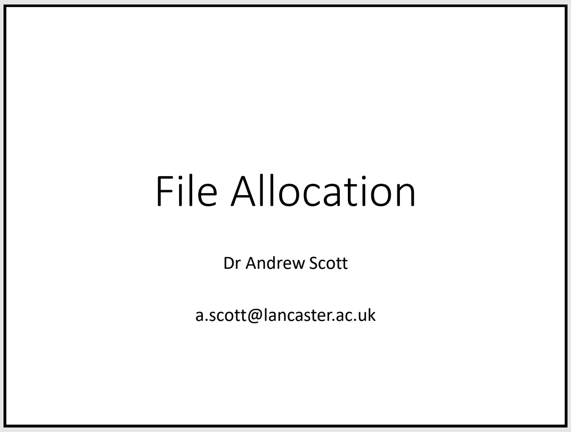
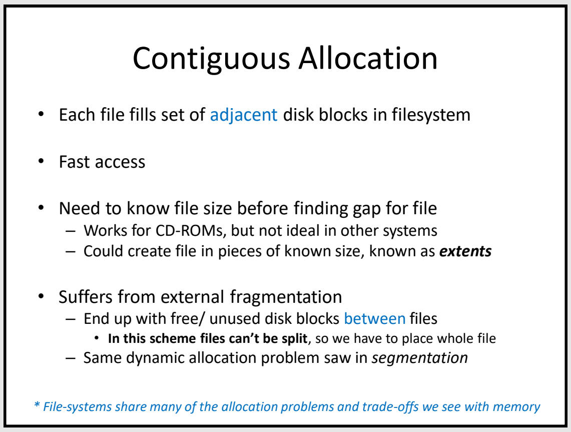


File Allocation

Dr Andrew Scott

a.scott@lancaster.ac.uk

1

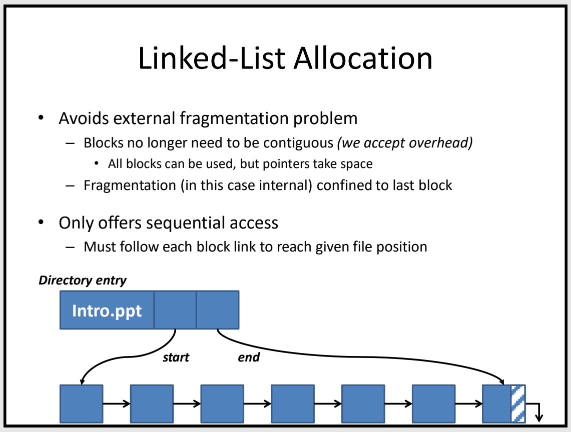


Contiguous Allocation

- Each file fills set of adjacent disk blocks in filesystem
- Fast access
- Need to know file size before finding gap for file
  - Works for CD-ROMs, but not ideal in other systems
  - Could create file in pieces of known size, known as ***extents***
- Suffers from external fragmentation
  - End up with free/ unused disk blocks between files
    - **In this scheme files can't be split**, so we have to place whole file
  - Same dynamic allocation problem saw in *segmentation*

** File-systems share many of the allocation problems and trade-offs we see with memory*

2



Linked-List Allocation

- Avoids external fragmentation problem
  - Blocks no longer need to be contiguous *(we accept overhead)*
    - All blocks can be used, but pointers take space
  - Fragmentation (in this case internal) confined to last block
- Only offers sequential access
  - Must follow each block link to reach given file position

Directory entry
Intro.ppt
start
end

4

Indexed Allocation

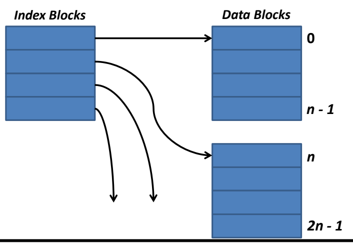
- Dedicated index block(s)
  - Quick access to any position/ address in file
    - Simple mapping to { Block, Offset }
- Can support ***holes* ***
  - i.e. we don't allocate space for 'empty', 0 filled blocks
  - Good for code storing data/ records at calculated file offsets
    - open(file); seek(file_position); write(record); close( );
- Three approaches:
  - linked, multi-level, and combined…

*** Note: holes are part of file – if we read from a hole we read 0s**

5

Indexed Allocation Example

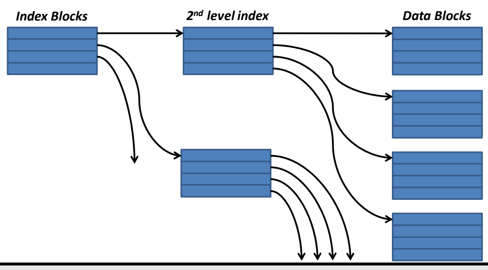
- Index block holds set of block addresses



6

Multi-Level Indexed Allocation

- Can be any number of levels deep



7

Indexed Allocation

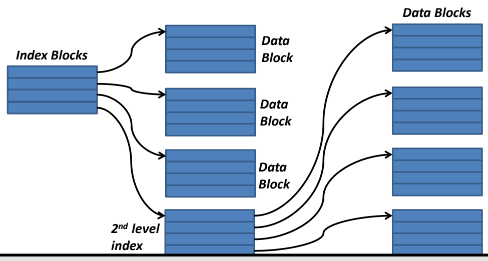
- Indexed (aka. Linked* allocation)
  - ✓ Simple and fast
  - ✗ Limited file size: only room for *n* block pointers in index block
- Multi-level
  - ✓ Supports large file sizes
  - ✓ Number of indexes can be tuned to max desired file size
    - But still fixed when file-system created
  - ✗ Access time increases with number of index levels
- ➢ Combined (*Unix approach*)
  - ✓ Good compromise: fast for small files
  - ✓ Access time slows as file size increases

** Don't confuse with Linked-List allocation*

8

Combined Allocation

- First entries are direct, single level entries
- Last entries are multi-level entries



9