Disk Allocation

> Access patterns, allocation - file caching

> Two basic diffs from mem mgt:
  correctness: crashes
  Performance: disk's nature

Access Patterns

> Sequential access
  File read/written in order
  Very common
  => much knowledge to OS (optimization)

> Random access
  Address some arbitrary block in file
  Harder to predict patterns

> Keyed index (Associative)
  Return data assoc. of key value - Database
    "higher level"
  Usually not in OS

Workloads

> File Characteristics
  \{ most files small \}
  \{ most of disk allocated to large files \}

  => keep per-file overheads low
  Good bandwidth for large files

> Common operations
  \{ read file => read i-node too \}
  \{ access files in same directory @ same time \}
  (e.g. make)
Free Space Mgmt

> Bit Maps
  array of bits, one per block
  perm. on disk, but keep in mem too (cache)

> Try to "co-allocate" "related" blocks
  when empty, easy
  when full, = time spent searching

> Partial Solution
  always keep some free at disk free
  (not available to users)

Strategies

> Progression
  simple ⇒ ...

> Keep in mind
  fragmentation & space/time
  ability to extend file & functionality
  access time & performance
  space & performance

<table>
<thead>
<tr>
<th>Time</th>
<th>few seeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>overhead of structures, fragmentation</td>
</tr>
<tr>
<td>Functionality</td>
<td>extend file</td>
</tr>
</tbody>
</table>
Allocate files like segmented memory \((\text{base} + \text{length})\)

- specify length at creation time
- find space by examining bit-map \((\text{first, best fit})\)
- inode/metadata: \(\text{base} + \text{length}\)

\[ \text{file 3} \
\]

+ easy to get at data \((\text{offset + base})\)
  sequential access efficient

- Fragmentation (which kind?)
  \(\Rightarrow\) solution: \underline{off-line compaction}
  Hard to predict requirements \((\text{size})\) at creation time
  \(\Rightarrow\) \(\text{res} \leftarrow \text{no stream off of the web}\)
  how to extend file?
**Extent-Based**

- Multiple contiguous regions / file
  - Meta-data: array of \(<ptr, len>\) pairs

  ![Diagram of extent-based file structure](image)

- \[ \text{fixed number} \]
  - How to find offset \(x\)?

- Files can grow over time; eases external fragmentation

- \(\text{fixed # of extents, frag. still an issue}\)

**Linked - Allocation**

- File kept as linked list of \text{fixed sized blocks}\n
  ![Diagram of linked file structure](image)

- Meta-data: pointer to first block

  - \(\text{e.g. TOPS-10, Alto}\)

- Easy to extend file; external frag. gone

- \(\text{random access? (impossible)}\)

- \(\text{sequential access}\)
  - Many seeks

- \(\text{slight space overhead per block}\)

- \(\text{internal fragmentation}\)
Variation: FAT Table

Beginning of disk: File Alloc for all files
1 entry / block
Linked list of entries / file

File A
File B
File C

FAT

```
0 1
| 0 | Free |
| 1 | Free |
| 2 | EOF  |
| 3 | EOF  |
| 4 | EOF  |
| 5 | Free |
| 6 | Free |
| 7 | Free |
```

Key: [Cache FAT in memory]

```
``Directory``
``A`` 1
``B`` 6
``C`` 9
```

+ w/ caching, better than linked list

- could have to read two blocks for each read
Indexed Allocation

> Meta-Data: Array of block pointers

+ No ext. frag.
supports random access

- lots of seeks for seq. access

Multi-Level Indexed Files (tree like)

++ very general
random access
large files
supported

-- many lookups = large files
still true: layout of file is important