File System Implementation.

Last class: FS APIs

Data Structures (on-disk)

Access Methods.

types: files, directories, symbolic links

low-level name

human name → low-level name

(number)

APIs:

\[
\text{open} \left(\text{"file"}, \text{O_RDONLY}\right)
\]

\[
\text{read} \left(\text{fd}, \text{buffer}, \text{size}\right)
\]

\[
\text{write} \left(\text{"\"}\right)
\]

\[
\text{fsync} \left(\text{fd}\right)
\]

\[
\text{seek} \left(\text{fd}, \ldots\right)
\]

\[
\text{close} \left(\text{fd}\right)
\]

open read write .. close().

<table>
<thead>
<tr>
<th>sched</th>
<th>VM</th>
<th>File System</th>
</tr>
</thead>
</table>

First: store user data \( \rightarrow \) DATA blocks

OS

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2 ( \sqrt{3} )</th>
<th>4 ( \sqrt{5} )</th>
<th>6</th>
<th>7 ( \sqrt{8} )</th>
<th>9</th>
<th>63</th>
</tr>
</thead>
</table>
5 | inode | data | bitmap | I | I | I | I | I | \( \cdot \) | \( \cdot \) | \( \cdot \) | \( \cdot \) | D |
0 | 4KB | 8KB | 16KB |

1 block = 4 KB \( \rightarrow \) 8 sectors
1 sector = 512 bytes

data blocks

Size of HDD = 64 \( \times \) 4 KB

\[ = 2 \times 2^6 \text{KB} = 256 \text{ KB} \]

Second: per-file metadata \( \rightarrow \) inode \( \rightarrow \) index node \( \rightarrow \) nodes, ...

\# blocks, \# links, size, type, owner, permissions, inode number, access time,

VSFS
Very Simple File System

Size of (inode) = 128 bytes
\( \rightarrow \) 256 bytes

data block 8

data block 21

data block 51
blocks
5 inodes → 9 inodes

1 block → \( \frac{4 \text{ KB}}{256} = \frac{2^{12}}{2^8} = 2^4 = 16 \)

⇒ 5 block ⇒ 16 × 5 = \( \text{80 inodes} \) ⇒ \( \text{80 files} \)

OS/FS → read block \( \textbf{8} \) ?

\( \text{block #} \)

8 × 4 KB = 32 KB = \( \frac{2^{15}}{2^{10}} \)

= \( 2^6 = \text{64} \) \( \text{sector #} \)

OS/FS → read inode \( \textbf{32} \) ?

Effect from inode start = 32 × 256 = \( 2^5 \times 2^8 \) = 8 KB

inode addr = 12 KB + 8 KB = 80 KB

sector # = \( \frac{80 \times 2^{10}}{512} \) = \( \text{40} \) \( \text{sector #} \)
Third:

need space for directories

→ just use inodes (type = directory)

→ contents

*: directory is just a file!

\[
\begin{array}{|c|c|}
\hline
\text{human name} & \text{inode#} \\
\hline
\vdots & 6 \\
\vdots & 2 \\
\text{bar.txt} & 10 \\
\hline
\end{array}
\]

\[ / 	ext{foo} \]

→ Data block for a directory.

Fourth:

need way to track FREE/ALLOCATED blocks

inode bitmap

data bitmap

\[
D_0 \quad D_1 \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad D_{55}
\]

\[
\begin{array}{ccccccccccc}
1 & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 1 & 1 & \ldots & \ldots & \ldots & 1 & 1 & 0
\end{array}
\]

(7 bytes)

(10 bytes)
Fifth: How big is the Fs?
   Where in the disk
   → type of Fs.
      ext2, ext3, ext4
   → How many inodes are in inode table?
      SUPER BLOCK.
Access Methods

1. Read
   read (fd, buf, size)
   bar.
   access the "data" blocks of bar.
   inode of bar.
   data bitmap
   inode of bar
   read data block of foo.
   readinode of root
   read "data" block of root
   read "super block" to get the inode # of the root dir.

<table>
<thead>
<tr>
<th>H-N</th>
<th>inode #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>foo</td>
<td>4</td>
</tr>
</tbody>
</table>

content of dir / (root)
2. Write `/foo/bar`

\[
\text{rc = write(fd, buf, size)};
\]

1. read inode of root(1)
2. read data of root(1)
3. read inode of foo.
4. read data of foo.
5. read inode of foo:bar

6. read data bitmap to find a free block.
7. Write to data bitmap for the corresponding data block.
8. Write to inode of bar. (addr of the new data block).
9. Write to the data block of bar
10. Write to inode of bar.
Page Cache
inode of 1
(inode of home)

fd

/foul/bar

f sync (fd)

1. create
2. write
3. f sync (fd), crash

bar

CS 537