File systems

Two levels in file system

+ user API (1)

+ block interface (2)

File system refers to layout of files

directories

i-nodes

Blocks within file are organized in a linked list style

Extent lists

(Block, len) → \[\ldots \square \square \square \ldots\]

<Block, len> → \[\square \ldots \square \square \square \ldots\]

A good way of representing the contents of a file.
File contents: (can be organized in different ways)

(i) Linked Lists (FAT)
(ii) Indirect Blocks
(iii) Extent List

Free space:

(i) Linked List
(ii) Bit-map
(iii) Extent List

Journaling, log structure

refers to writing records in to log helps to recover from crashes

Security

permission matrix

<table>
<thead>
<tr>
<th></th>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>r--</td>
<td>r--</td>
<td>r--</td>
</tr>
<tr>
<td>User 2</td>
<td>r--</td>
<td>rw-</td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>rw-</td>
<td>rwx</td>
<td></td>
</tr>
<tr>
<td>User 4</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

Systems have two ways to handle these permissions

(i) User + Group

(ii) ACL's
(i) **User + Group**

Create a set of groups and assign users to a group.

(ii) **Access control lists**. A simple list of rules we can transform an matrix to a list of rules.

**Capabilities & Permissions**

Least privilege.

**Code Injection** (stack smashing)

```
{ chan buf[80]
  gets (buf)
  ... do something
  }
```

If `gets` retrieves more than 80 bytes, it will overwrite in the stack.

Address space layout randomization can solve these attacks to some extent.
For some new processors we can have an extra bit called NX bit which can be set to prevent a page from executing. So for a stack page NX bit is set then processor cannot execute code from that page.

```
stack

A4 → ret · ·
A3 → mov ax, bx
A2 → ret
A1 → add 5, ax
```