Chapter 12
Operating System Design

12.1 The nature of the design problem
12.2 Interface design
12.3 Implementation
12.4 Performance
12.5 Project management
12.6 Preads in operating system design

Paradigms (1)

Algorithmic code

Paradigms (2)

Event-driven code

Implementation

One possible design for a modern layered operating system
Naming

Directories are used to map external names onto internal names.

Static Versus Dynamic Structures

Searching a static table for a pid:

```c
found = 0;
for (p = &proc_table[0]; p < &proc_table[PROC_TABLE_SIZE]; p++) {
    if (p->proc_pid == pid) {
        found = 1;
        break;
    }
}
```

Hiding the Hardware (1)

```
#include "config.h"
init() {
    #if (CPU == PENTIUM)
    /* Pentium initialization here. */
    #endif

    #if (CPU == ULTRASPARC)
    /* UltraSPARC initialization here. */
    #endif

    CPU-dependent conditional compilation
```

Hiding the Hardware (2)

```
#include "config.h"
#if (WORD_LENGTH == 32)
typedef int Register;
#endif

#if (WORD_LENGTH == 64)
typedef long Register;
#endif

Register R0, R1, R2, R3;
```

Word-length dependent conditional compilation
Space-Time Trade-offs (1)

```c
#define BYTE_SIZE 8
int bit_count(int byte) {
    int i, count = 0;
    for (i = 0; i < BYTE_SIZE; i++)
        if ((byte >> i) & 1) count++;
    return(count);
}

/* A byte contains 8 bits */
/* Count the bits in a byte. */
/* loop over the bits in a byte */
/* if this bit is a 1, add to count */
/* return sum */
```

A procedure to count the 1 bits in a byte

Space-Time Trade-offs (2)

```c
/* Macro to add up the bits in a byte and return the sum. */
#define bit_count(b) ((b&1) + ((b>>1)&1) + ((b>>2)&1) + ((b>>3)&1) +
                    ((b>>4)&1) + ((b>>5)&1) + ((b>>6)&1) + ((b>>7)&1))

/* Macro to look up the bit count in a table. */
char bits[256] = {0, 1, 1, 2, 2, 3, 1, 2, 2, 3, 3, 4, 1, 2, 2, 3, 3, 3, 3, ...
#define bit_count(b) (int)bits[b]
```

(b) Macro to count the bytes
(c) Macro to look up the count

Space-Time Trade-offs (3)

(a) Part of an uncompressed image with 24 bits per pixel
(b) Same part compressed with GIF, 8 bits per pixel
(c) The color palate

Caching

<table>
<thead>
<tr>
<th>Path</th>
<th>I-node number</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr</td>
<td>6</td>
</tr>
<tr>
<td>/usr/ast</td>
<td>26</td>
</tr>
<tr>
<td>/usr/ast/mbox</td>
<td>60</td>
</tr>
<tr>
<td>/usr/ast/books</td>
<td>92</td>
</tr>
<tr>
<td>/usr/bal</td>
<td>45</td>
</tr>
<tr>
<td>/usr/bal/paper.ps</td>
<td>85</td>
</tr>
</tbody>
</table>

Part of an i-node cache
Software team Structure

<table>
<thead>
<tr>
<th>Title</th>
<th>Duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chief programmer</td>
<td>Performs the architectural design and writes the code</td>
</tr>
<tr>
<td>Copilot</td>
<td>Helps the chief programmer and serves as a sounding board</td>
</tr>
<tr>
<td>Administrator</td>
<td>Manages the people, budget, space, equipment, reporting, etc.</td>
</tr>
<tr>
<td>Editor</td>
<td>Writes the documentation, which must be written by the chief programmer</td>
</tr>
<tr>
<td>Secretaries</td>
<td>The administrator and editor each need a secretary</td>
</tr>
<tr>
<td>Program clerk</td>
<td>Maintains the code and documentation archives</td>
</tr>
<tr>
<td>Toolsmith</td>
<td>Provides any tools the chief programmer needs</td>
</tr>
<tr>
<td>Tester</td>
<td>Tests the chief programmer’s code</td>
</tr>
<tr>
<td>Language lawyer</td>
<td>Part timer who can advise the chief programmer on the language</td>
</tr>
</tbody>
</table>

Mills’ proposal for populating a 10-person chief programmer team

The Role of Experience (1)

Traditional software design progresses in stages

The Role of Experience (2)

- Alternative design produces a working system
  - that does nothing starting on day 1