Chapter 12

Operating System Design

Paradigms (1)

```
main( )
{
        int ... ;

        init( );
        do_something( );
        read(...);
        do_something_else( );
        write(...);
        keep_going( );
        exit(0);
}
```

Algorithmic code

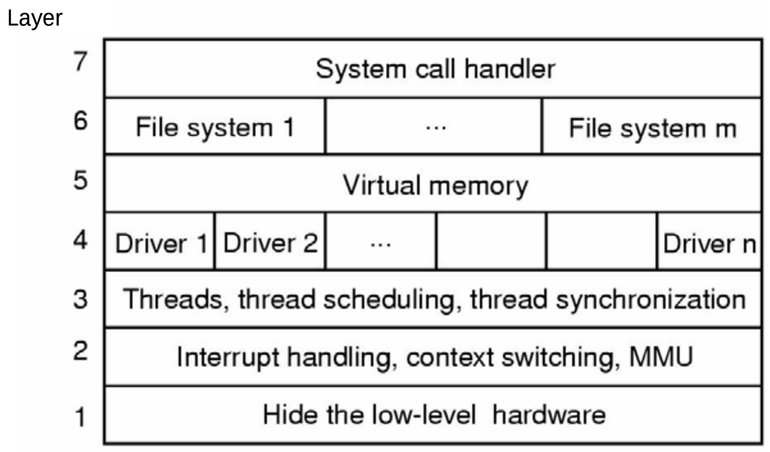
Paradigms (2)

```
main( )
{
        mess_t msg;

        init( );
        while (get_message(&msg)) {
                switch (msg.type) {
                        case 1: ... ;
                        case 2: ... ;
                        case 3: ... ;
                }
        }
}
```

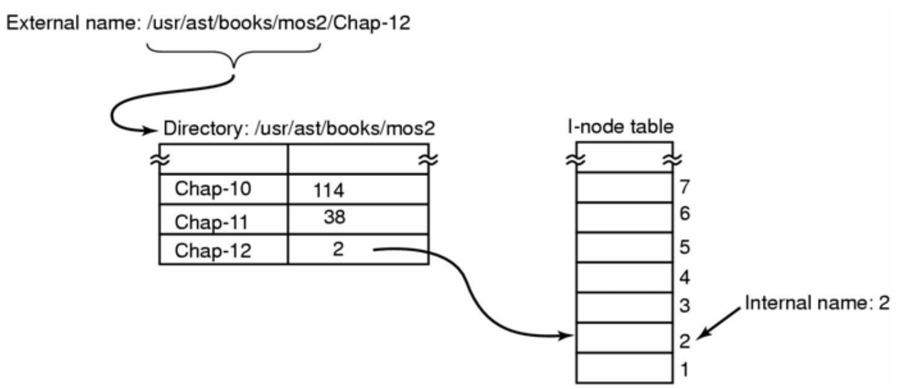
Event-driven code

Implementation


Layer
7
System call handler
6
File system 1
...
File system m
5
Virtual memory
4
Driver 1
Driver 2
...
Driver n
3
Threads, thread scheduling, thread synchronization
2
Interrupt handling, context switching, MMU
1
Hide the low-level hardware

One possible design for a modern layered operating system

Naming


External name: /usr/ast/books/mos2/Chap-12
Directory: /usr/ast/books/mos2
I-node table
Chap-10 114
Chap-11 38
Chap-12 2
7
6
5
4
3
2
1
Internal name: 2

Directories are used to map external names onto internal names

Static Versus Dynamic Structures

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

Searching a static table for a pid

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)

```
#define BYTE_SIZE 8                         /* A byte contains 8 bits */
int bit_count(int byte)
{                                           /* Count the bits in a byte. */
    int i, count = 0;
    for (i = 0; i < BYTE_SIZE; i++)         /* loop over the bits in a byte */
        if ((byte >> i) & 1) count++;       /* if this bit is a 1, add to count */
    return(count);                          /* return sum */
}
```
(a)

A procedure to count the 1 bits in a byte

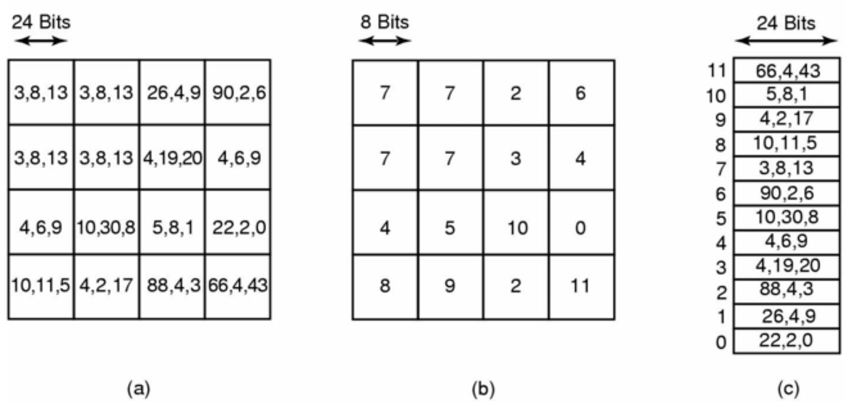
Space-Time Trade-offs (2)

```
/*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)
```
(b)

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

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

Space-Time Trade-offs (3)



(a)

3,8,13	3,8,13	26,4,9	90,2,6
3,8,13	3,8,13	4,19,20	4,6,9
4,6,9	10,30,8	5,8,1	22,2,0
10,11,5	4,2,17	88,4,3	66,4,43

(b)

7	7	2	6
7	7	3	4
4	5	10	0
8	9	2	11

(c)

11	66,4,43
10	5,8,1
9	4,2,17
8	10,11,5
7	3,8,13
6	90,2,6
5	10,30,8
4	4,6,9
3	4,19,20
2	88,4,3
1	26,4,9
0	22,2,0

(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

Path	I-node number
/usr	6
/usr/ast	26
/usr/ast/mbox	60
/usr/ast/books	92
/usr/bal	45
/usr/bal/paper.ps	85

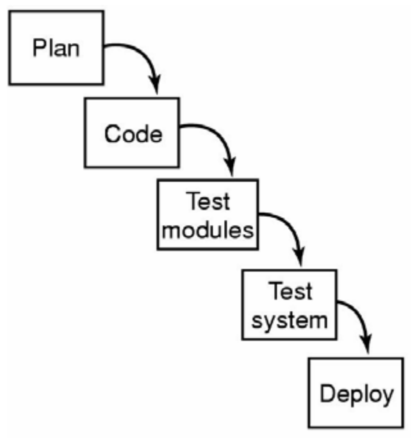
Part of an i-node cache

Software team Structure

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

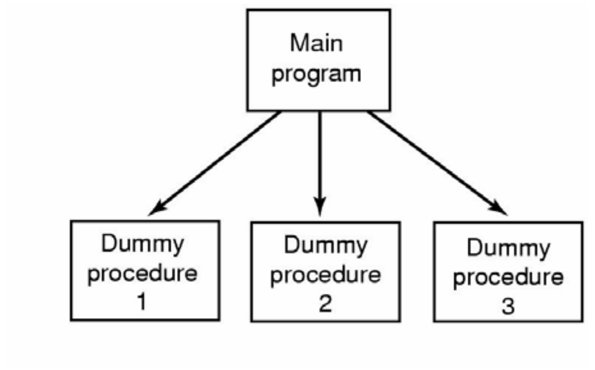
Mills' proposal for populating a 10-person chief programmer team

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