MODERN OPERATING SYSTEMS

Third Edition

ANDREW S. TANENBAUM

Chapter 13
Operating System Design

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Operating System Goals

- Define abstractions.
- Provide primitive operations.
- Ensure isolation.
- Manage the hardware.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Difficulties Designing Operating Systems (1)

- Operating systems have become extremely large programs
- Must deal with concurrency
- Have to deal with potentially hostile users
- Many users want to share some of their information and resources with selected other users

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Difficulties Designing Operating Systems (2)

- Operating systems live for a very long time
- Designers need to provide for considerable generality
- Systems are generally designed to be portable
- Operating systems need to be backward compatible with some previous operating system

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Guiding Principles for Interface Design

- Simplicity
  Perfection is reached not when there is no longer anything to add, but when there is no longer anything to take away. (Antoine de St. Exupery)
- Completeness
  Everything should be as simple as possible, but no simpler. (Albert Einstein)
- Efficiency
  If a feature or system call cannot be implemented efficiently, it is probably not worth having (Tanenbaum)

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Execution Paradigms

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

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

(a)

```
main( )
{
        mess_t msg;

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

(b)

Figure 13-1. (a) Algorithmic code. (b) Event-driven code.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

System Structure

- Layered Systems
- Exokernels
- Microkernel-Based Client-Server Systems
- Extensible Systems
- Kernel Threads

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Layered Systems

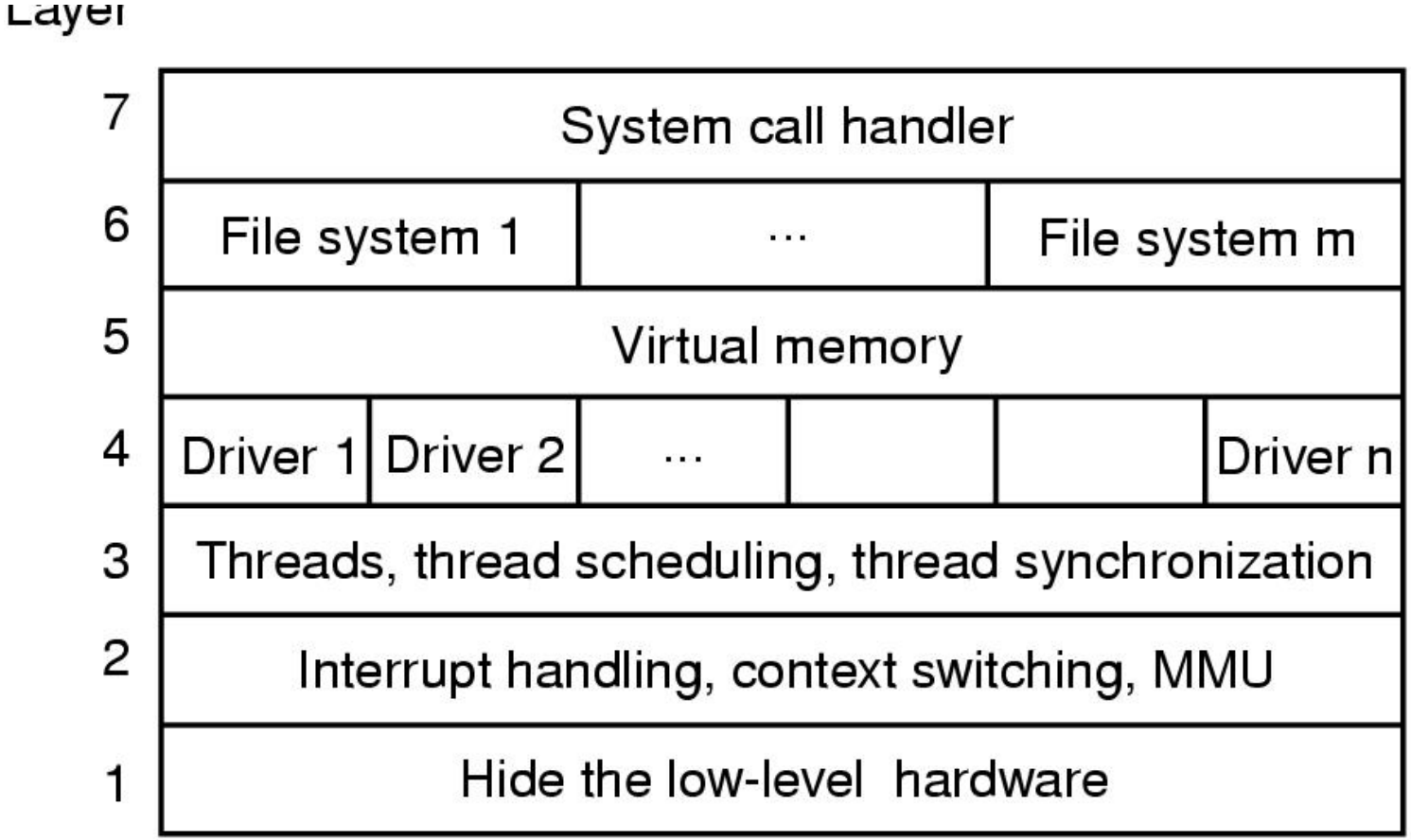

Figure 13-2. One possible design for a modern layered operating system.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Microkernel-Based Client-Server Systems

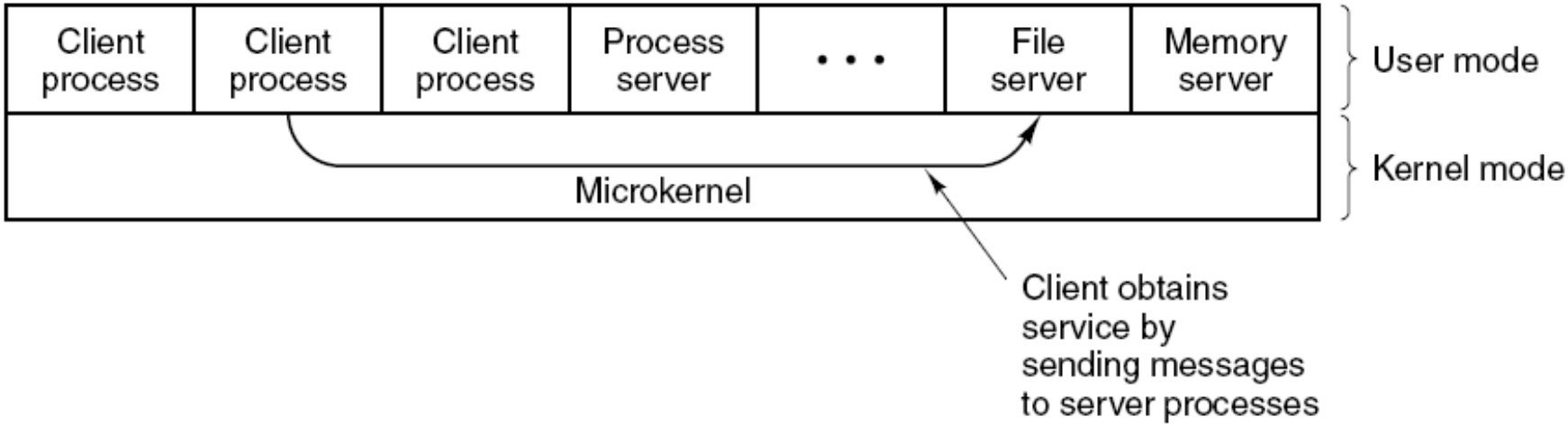

Figure 13-3. Client-server computing based on a microkernel.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Naming

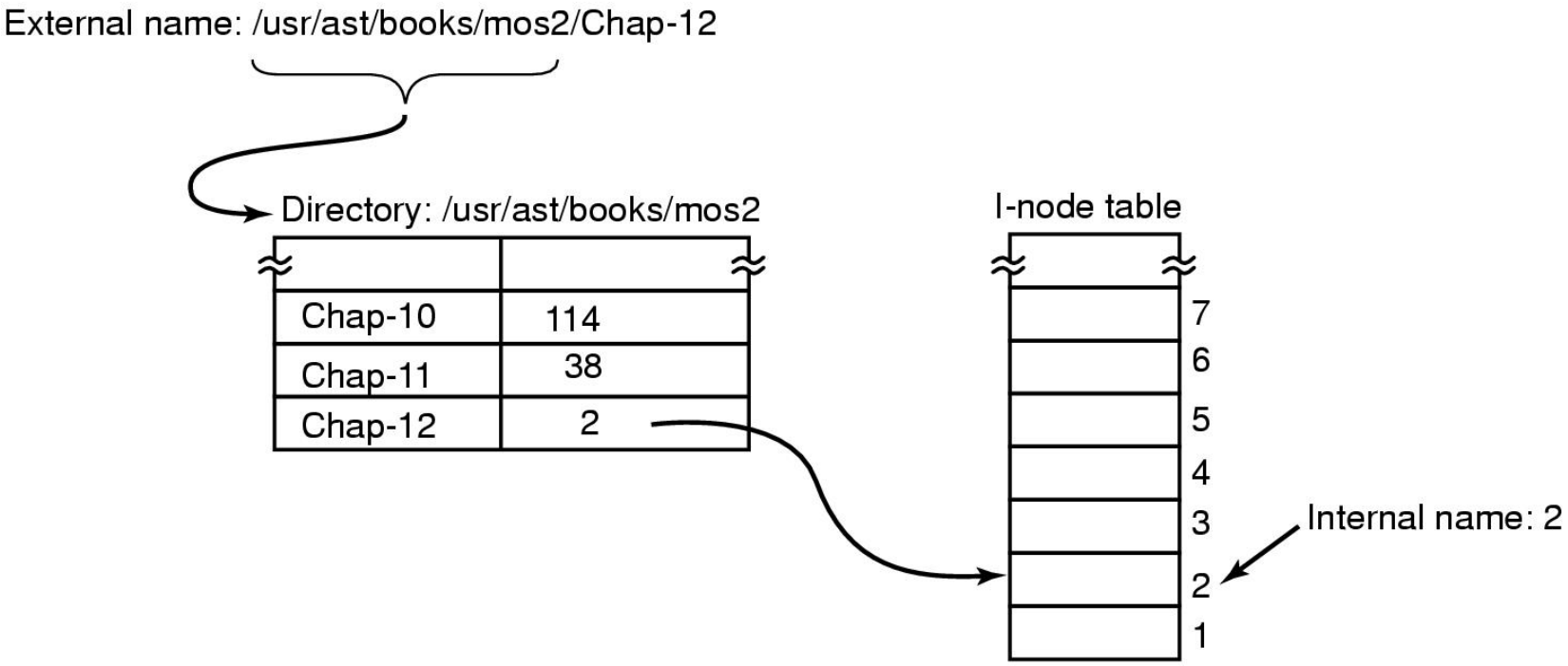

Figure 13-4. Directories are used to map external names onto internal names.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Binding Time

- Early binding
  - Simple
  - but less flexible
- Late binding
  - More complicated
  - but more flexible

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

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;
        }
}
```

Figure 13-5. Code for searching the process table for a given PID.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Useful Techniques

- Hiding the Hardware
- Indirection
- Reusability
- Reentrancy
- Brute Force
- Check for Errors First

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Hiding the Hardware (1)

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

Figure 13-6. (a) CPU-dependent conditional compilation. (

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Hiding the Hardware (2)

```
#include “config.h”

init( )

#if (CPU = PENTIUM)
…
#endif

#if (CPU = SPARC)
…
#endif
```

```
#include “config.h”

#if (WORD_LENGTH == 32)
typedef int register
#endif

#if (WORD_LENGTH == 64)
typedef long register
#endif
```

Figure 13-6. (b) Word-length dependent conditional compilation.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Performance

- Why Are Operating Systems Slow?
- What Should Be Optimized?
- Space-Time Trade-offs
- Caching
- Hints
- Exploiting Locality
- Optimize the Common Case

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

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)

```
/*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)

Figure 13-7. (a) A procedure for counting bits in a byte. (b) A macro to count the bits. (c) Table look up

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Space-Time Trade-offs (2)

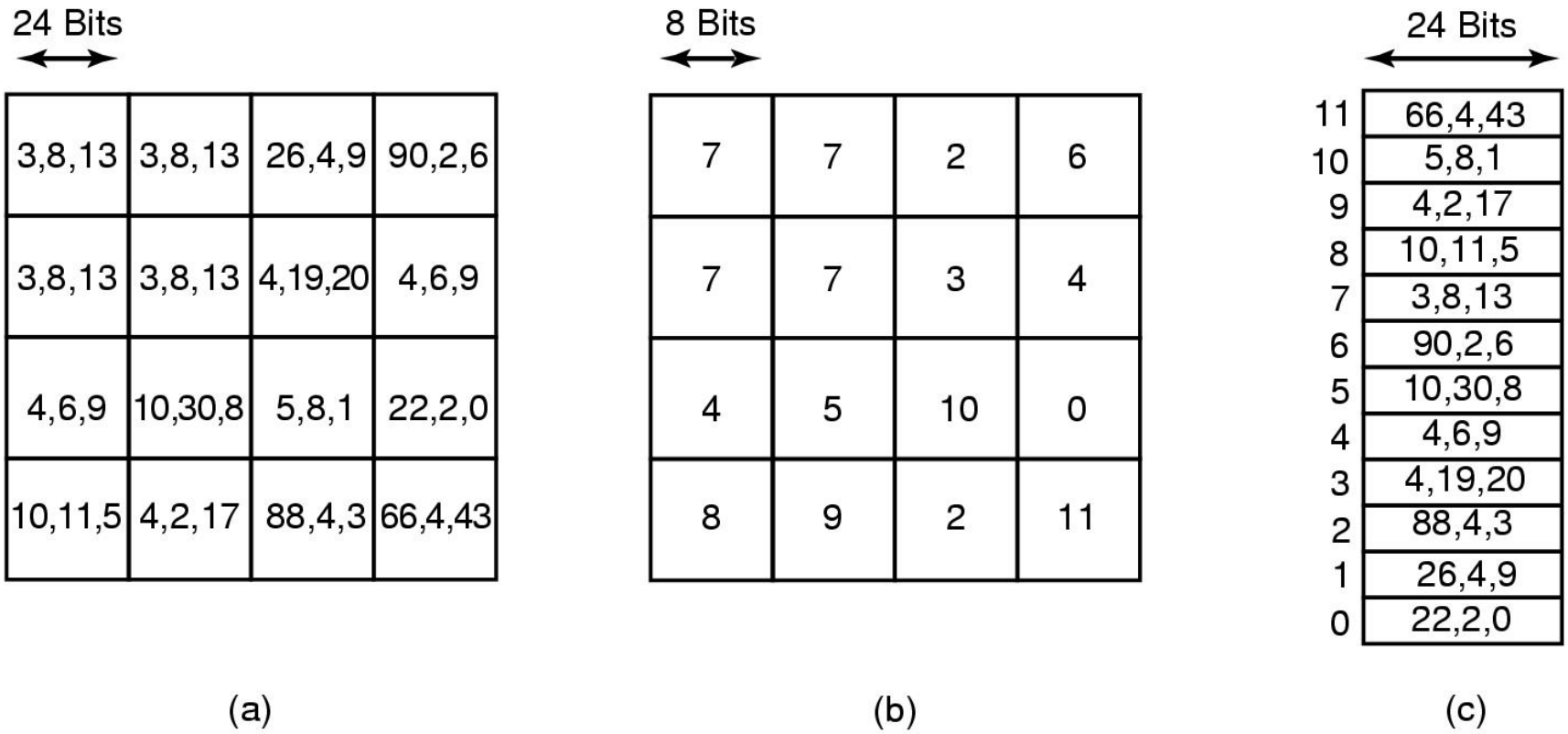

Figure 13-8. (a) Part of an uncompressed image with 24 bits per pixel. (b) With a palette

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Caching (1)

To look up */usr/ast/mbox* (Fig. 4-35) requires the following disk accesses:

- Read the i-node for the root directory (i-node 1).
- Read the root directory (block 1).
- Read the i-node for */usr* (i-node 6).
- Read the */usr* directory (block 132).
- Read the i-node for */usr/ast* (i-node 26).
- Read *the /usr/ast* directory (block 406).

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Caching (2)

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

Figure 13-9. Part of the i-node cache for Fig. 4-35.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Project Management
The Mythical Man Month

- Large project design
  - 1/3 Planning
  - 1/6 Coding
  - 1/4 Module testing
  - 1/4 System testing
- People and time not interchangeable
  - Work cannot be fully parallelized
  - Work must be partitioned into large numbers of modules
  - Debugging is highly sequential

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

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

Figure 13-10. Mills' proposal for populating a 10-person chief programmer team.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

The Role of Experience

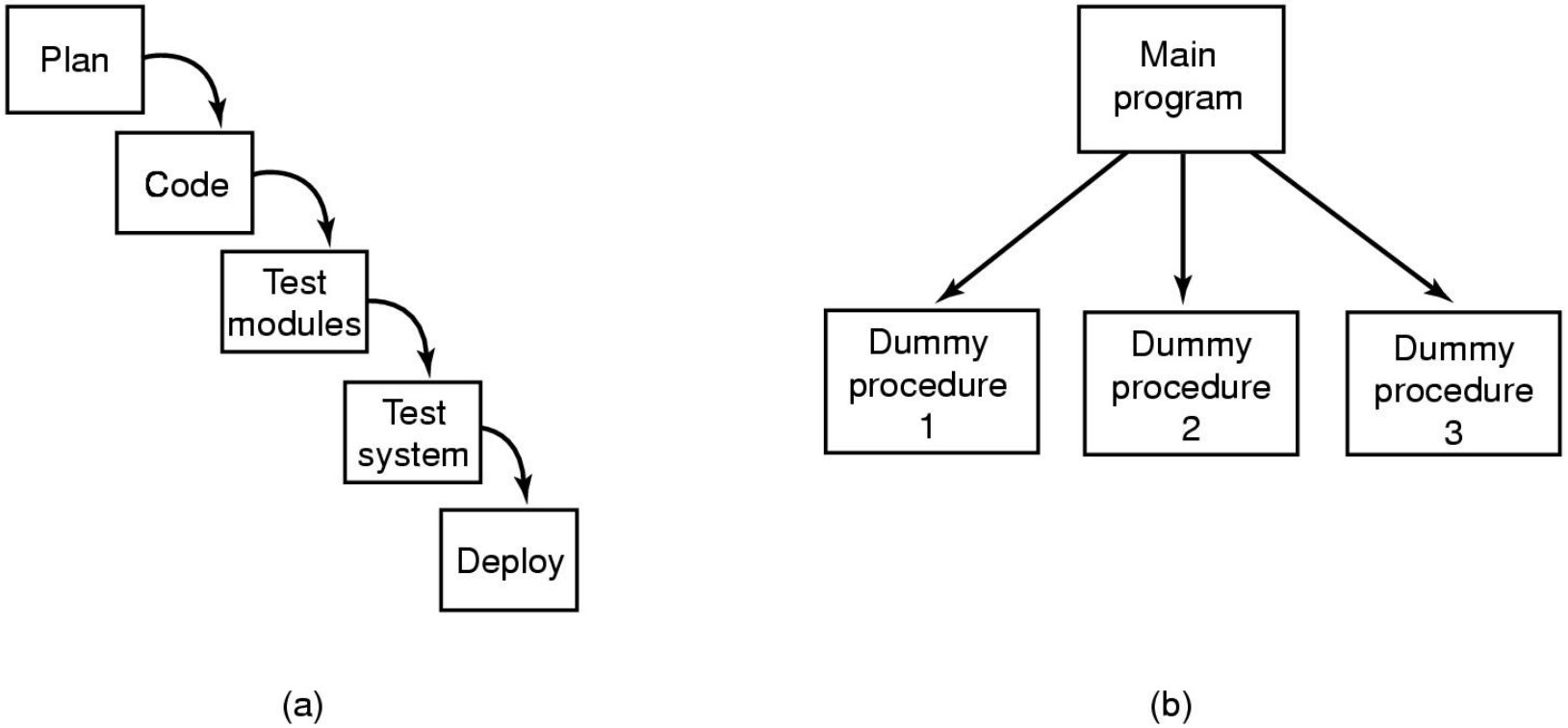

Figure 13-11. (a) Traditional software design progresses in stages. (b) Alternative design produces a working system (that does nothing) starting on day 1.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Trends in Operating Systems

- Virtualization
- Multicore Chips
- Large Address Space Operating Systems
- Networking
- Parallel and Distributed Systems
- Multimedia
- Battery-Powered Computers
- Embedded Systems
- Sensor Nodes

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**

Virtualization

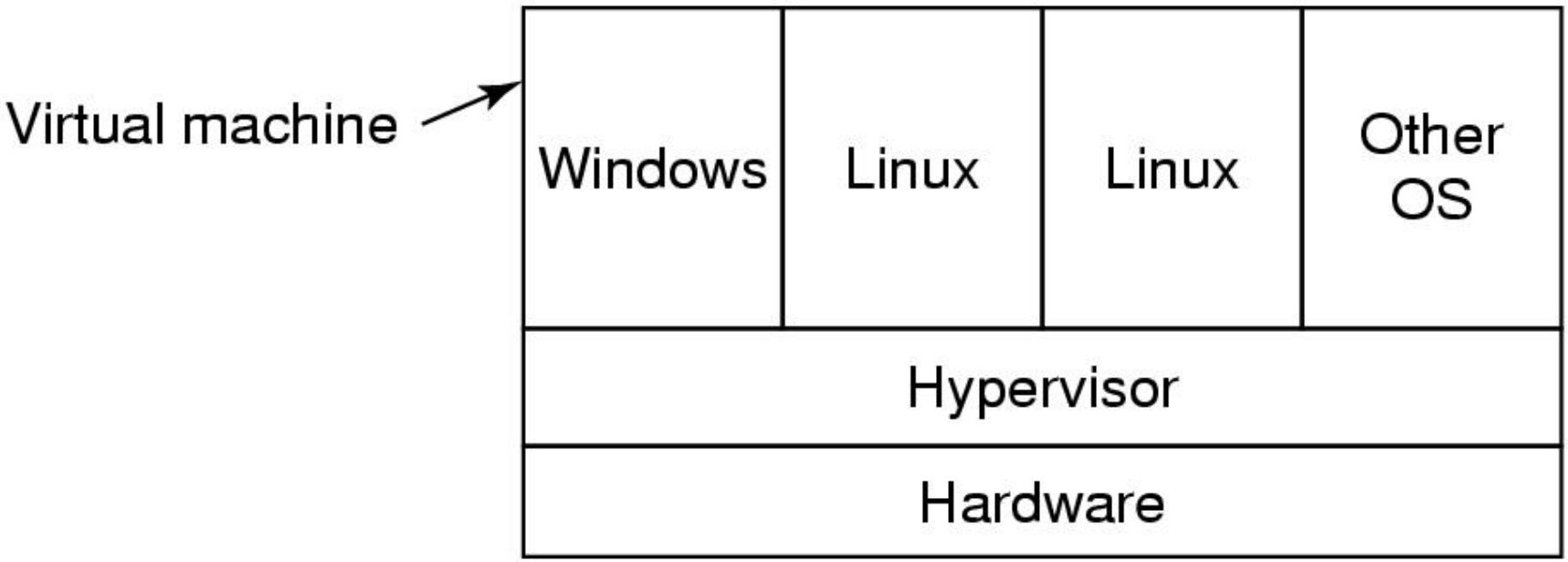

Figure 13-12. A hypervisor running four virtual machines.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**