COT 4600 Operating Systems Fall 2009

Dan C. Marinescu

Office: HEC 439 B

Office hours: Tu-Th 3:00-4:00 PM

Lecture 7

- Last time:
  - 2. Interpreters
  - 3. Communication Links
  - Internet or what is behind the abstractions…
- Today:
  - Naming in computing systems
- Next Time
  - Case Study: Unix File System

Announcements:

- no office hours on Thursday, September 17.
- Phase 1 of the project is due on Thursday, September 17
- HW 2 is due on Thursday September 24.

Naming

- The three abstractions (memory, interpreters, communication links manipulate objects identified by name.
- How could object A access object B:
  - Make a copy of object B and include it in A ➔ use by value
    - Safe → there is a single copy of B
    - How to implement sharing of object B?
  - Pass to A the means to access B using its name → use by reference
    - Not inherently safe → both A and C may attempt to modify B at the same time. Need some form of concurrency control.

Binding and indirection

- Indirection → decoupling objects from their physical realization through names.
- Names allow the system designer to:
  1. organize the modules of a system and to define communication patterns among them
  2. defer for a later time
     - to create object B referred to by object A
     - select the specific object A wishes to use
- Binding → linking the object to names. Examples:
  - A compiler constructs
    - a table of variables and their relative address in the data section of the memory map of the process
    - a list of unsatisfied external references
  - A linker binds the external references to modules from libraries

Generic naming model

- Naming scheme → strategy for naming. Consists of:
  - Name space → the set of acceptable names; the alphabet used to select the symbols from and the syntax rules.
  - Universe of values → set of objects/values to be named
  - Name mapping algorithm → resolves the names, establishes a correspondence between a name and an object/value
  - Context → the environment in which the model operates.
    - Example: searching for John Smith in the White Pages in Orlando (one context) or in Tampa (another context).
    - Sometimes there is only one context → universal name space; e.g., the SSNs.
    - Default context

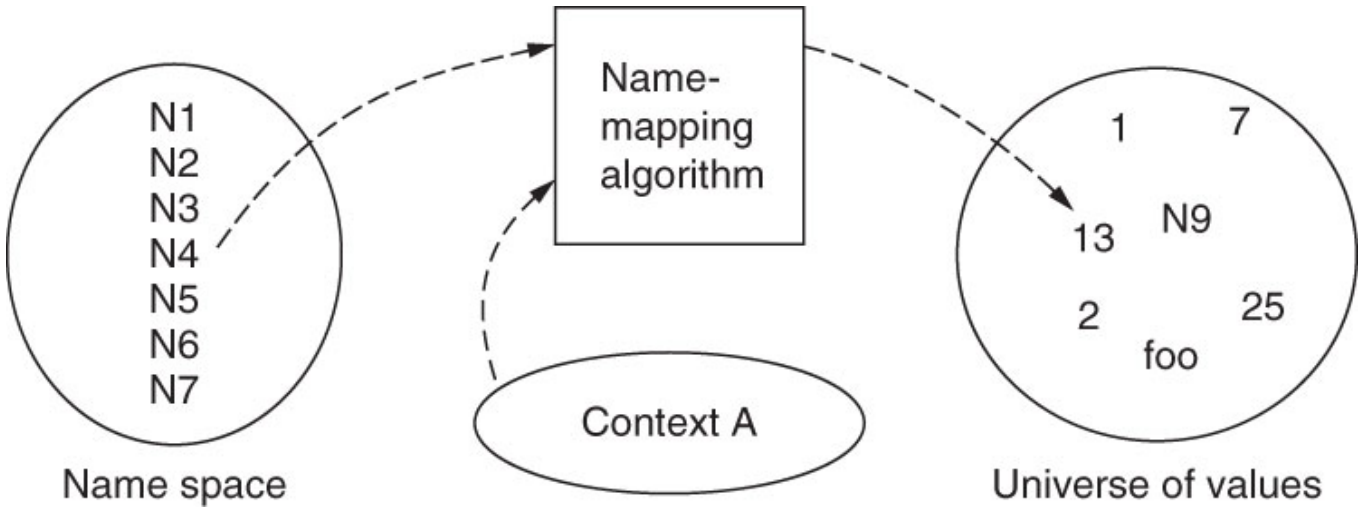


Figure 2.10 from the textbook

Operations on names in the abstract model

- Simple models: value ← RESOLVE (name, context)
  - The interpreter:
    - Determines the version of the RESOLVE (which naming scheme is used)
    - Identifies the context
    - Locates the object
  - Example: the processor
- Complex models support:
  - creation of new bindings: status ← BIND(name, value, context)
  - deletion of old bindings: status ← UNBIND(name, value)
  - enumeration of name space: list ← ENUMERATE(context)
  - comparing names status: result ← COMPARE(name1,name2)

Name mapping

- Name to value mapping
  - One-to-One → the name identifies a single object
  - Many-to-One → multiple names identify one objects (aliasing)
  - One-to-Many → multiple objects have the same name even in the same context.
- Stable bindings → the mapping never change. Examples:
  - Social Security Numbers
  - CustomerId for customer billing systems

Name-mapping algorithms

1. Table lookup
   1. Phone book
   2. Port numbers → a port the end point of a network connection
2. Recursive lookup:
   1. File systems – path names
   2. Host names – DNS (Domain Name Server)
   3. Names for Web objects - URL – (Universal Resource Locator)
3. Multiple lookup → searching through multiple contexts
   1. Libraries
   2. Example: the classpath is the path that the Java runtime environment searches for classes and other resource files

1. Table lookup

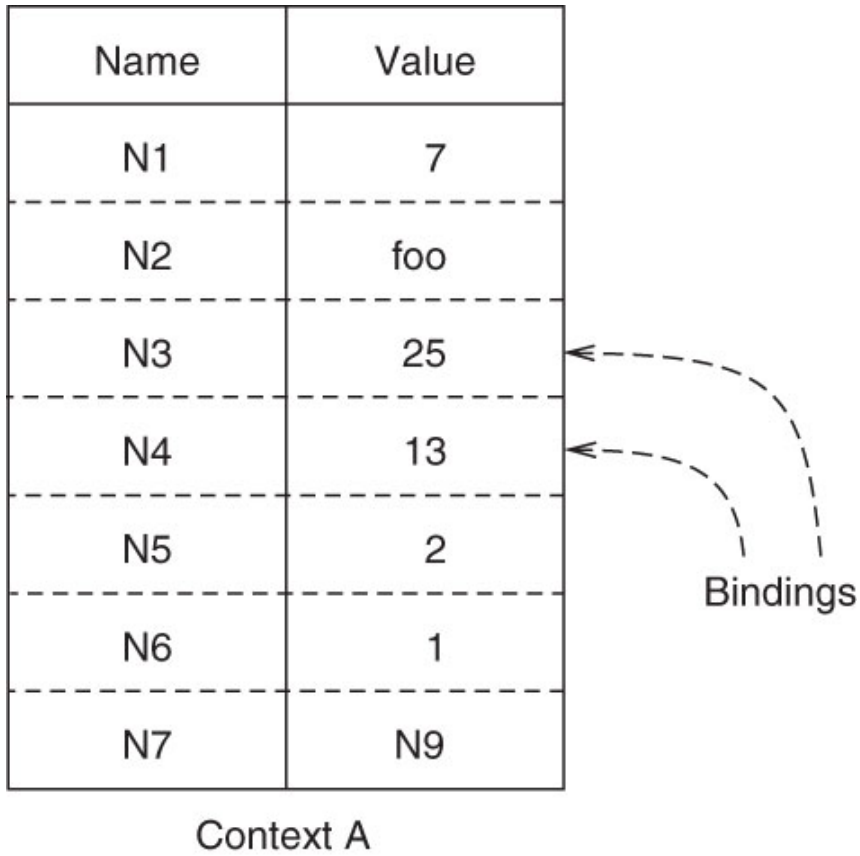


Figure 2.11 from the textbook

How to determine the context

- Context references:
  - Default → supplied by the name resolver
    - Constant → built-in by the name resolver
      - Processor registers (hardwired)
      - Virtual memory (the page table register of an address space)
    - Variable → supplied by the current environment
      - File name (the working directory)
  - Explicit → supplied by the object requesting the name resolution
    - Per object
      - Looking up a name in the phone book
    - Per name → each name is loaded with its own context reference (qualified name).
      - URL
      - Host names used by DNS

Dynamic and multiple contexts

- Context reference static/dynamic.
  - Example: the context of the “help” command is dynamic, it depends where you are the time of the command.
- A message is encapsulated (added a new header,) as flows down the protocol stack:
  - Application layer (application header understood only in application context)
  - Transport layer (transport header understood only in the transport context)
  - Network layer (network header understood only in the network context)
  - Data link layer (data link header understood only in the data link context)

2. Recursive name resolution

- Contexts are structured and a recursion is needed for name resolution.
- Root → a special context reference - a universal name space
- Path name → name which includes an explicit reference to the context in which the name is to be resolved.
  - Example: first paragraph of page 3 in part 4 of section 10 of chapter 1 of book “Alice in Wonderland.”
  - The path name includes multiple components known to the user of the name and to name solver
  - The least element of the path name must be an explicit context reference
- Absolute path name → the recursion ends at the root context.
- Relative path name → path name that is resolved by looking up its mot significant component of the path name

Example

- AliceInWonderland.Chapter1.Section10.Part4.Page3.FirstParagraph

  Most significant ← → Least significant

3. Multiple lookup

- Search path → a list of contexts to be searched

  Example: the classpath is the path that the Java runtime environment searches for classes and other resource files
- User-specfic search paths → user-specific binding
- The contexts can be in concentric layers. If the resolver fails in a inner layer it moves automatically to the outer layer.
- Scope of a name → the range of layers in which a name is bound to the same object.

Comparing names

- Questions
  - Are two names the same? → easy to answer
  - Are two names referring to the same object (bound to the same value)? → harder; we need the contexts of the two names.
  - If the objects are memory cells are the contents of these cells the same?

Name discovery

- Two actors:
  - The exporter → advertizes the existence of the name.
  - The prospective user → searches for the proper advertisement. Example: the creator of a math library advertizes the functions.
- Methods
  - Well-known names
  - Broadcasting
  - Directed query
  - Broadcast query
  - Introduction
  - Physical randezvoue

Computer System Organization

- Operating Systems (OS) → software used to
  - Control the allocation of resources (hardware and software)
  - Support user applications
  - Sandwiched between the hardware layer and the application layer
- OS-bypass: the OS does not hide completely the hardware from applications. It only hides dangerous functions such as
  - I/O operations
  - Management function
- Names → modularization

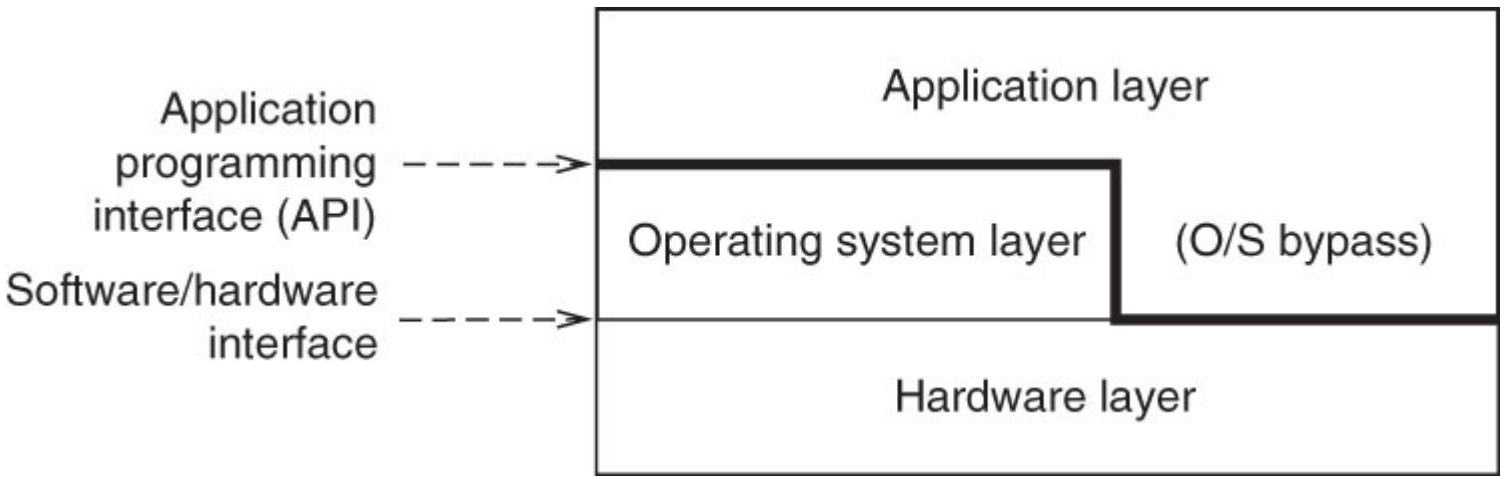


Figure 2.16 from the textbook

A. The hardware layer

- Modules representing each of the three abstractions (memory, interpreter, communication link) are interconnected by a bus.
- The bus → a broadcast communication channel, each module hears every transmission.
  - Control lines
  - Data lines
  - Address lines
- Each module
  - is identified by a unique address
  - has a bus interface
- Modules other than processors need a controller.

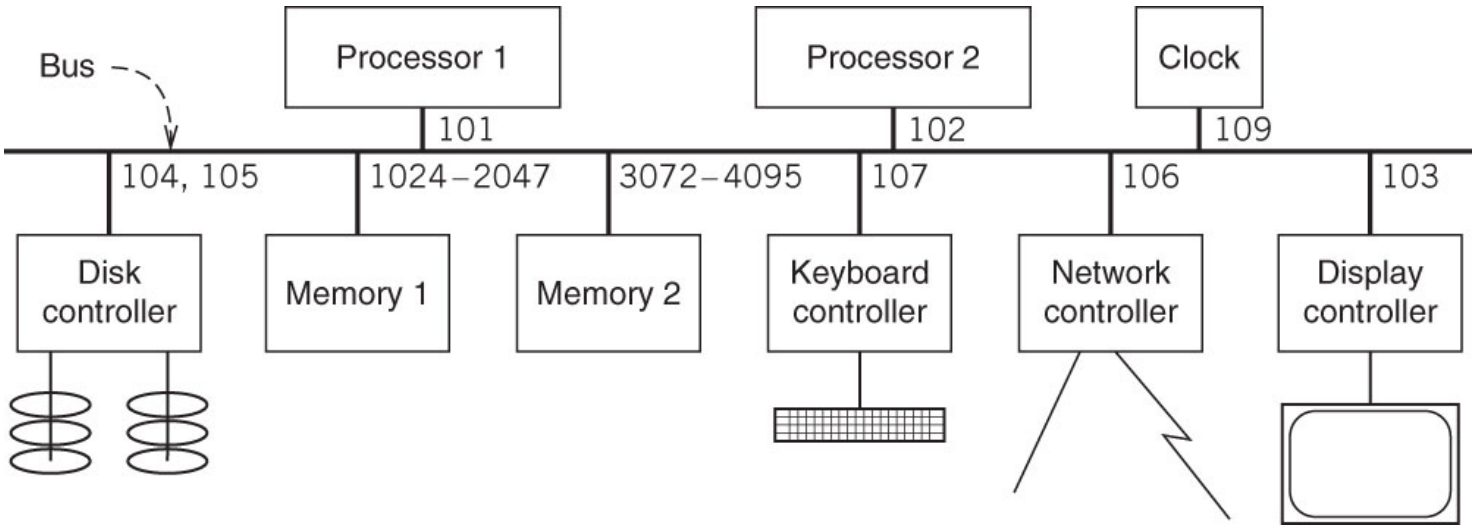


Figure 2.17 from the textbook

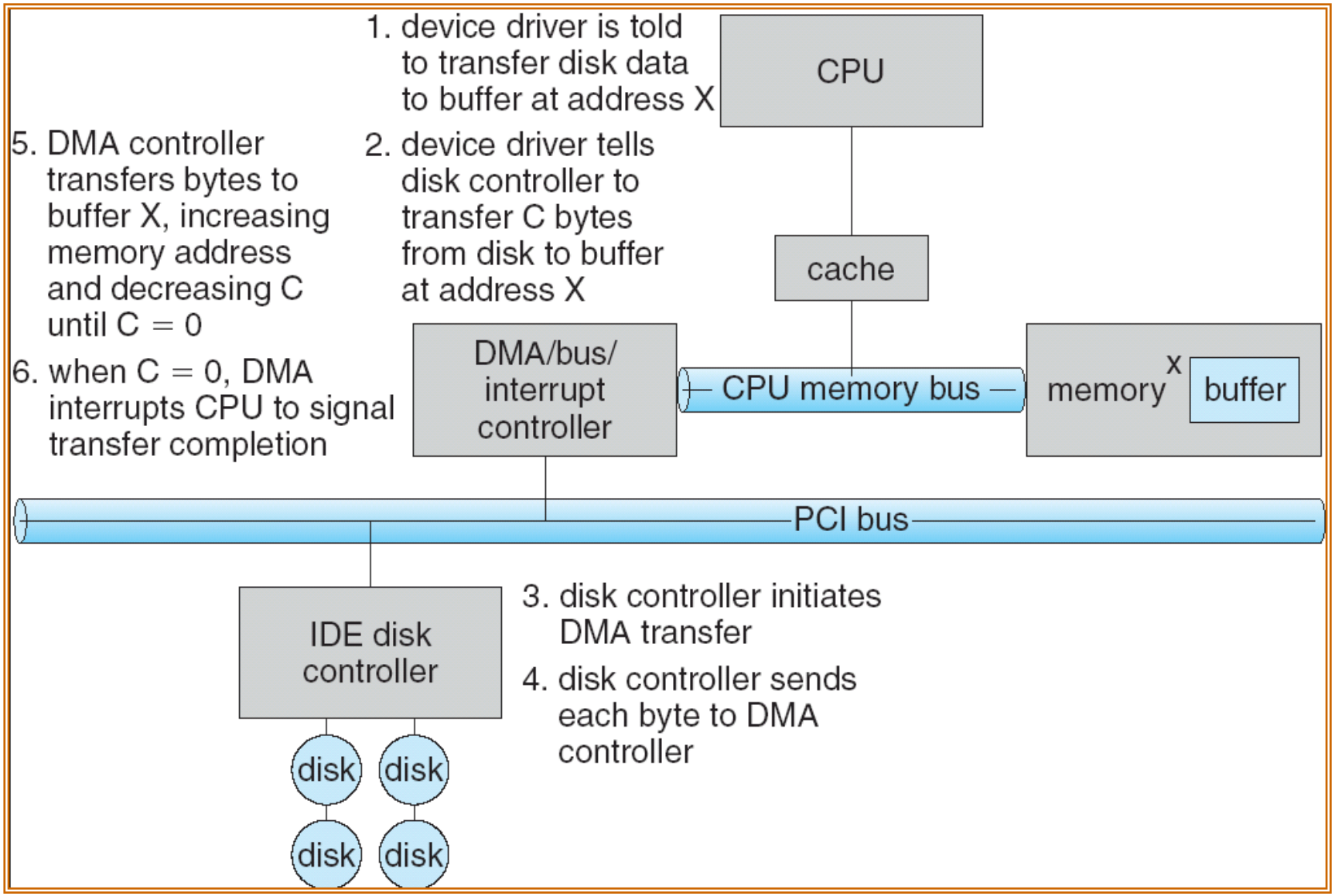
Bus sharing and optimization

- Communication → broadcast
- Arbitration protocol → decide which module has the control of the bus. Supported by hardware:
  - a bus arbiter circuit
  - distributed among interfaces – each module has a priority
  - daisy chaining
- Split-transaction → a module uses the arbitration protocol to acquire control of the bus
- Optimization:
  - hide the latency of I/O devices
    - Channels → dedicated processors capable to execute a channel program (IBM)
    - DMA (Direct Memory Access)
  - Support transparent access to files:
    - Memory Mapped I/O

Optimization

- Direct Memory Access (DMA):
  - supports direct communication between processor and memory; the processor provides the disk address of a block in memory where data is to be read into or written from.
  - hides the disk latency; it allows the processor to execute a process while data is transferred
- Memory Mapped I/O:
  - LOAD and STORE instructions access the registers and buffers of an I/O module
    - bus addresses are assigned to control registers and buffers of the I/O module
    - the processor maps bus addresses to its own address space (registers)
  - Supports software functions such as UNIX mmap which map an entire file.
- Swap area: disk image of the virtual memory of a process.

DMA Transfer

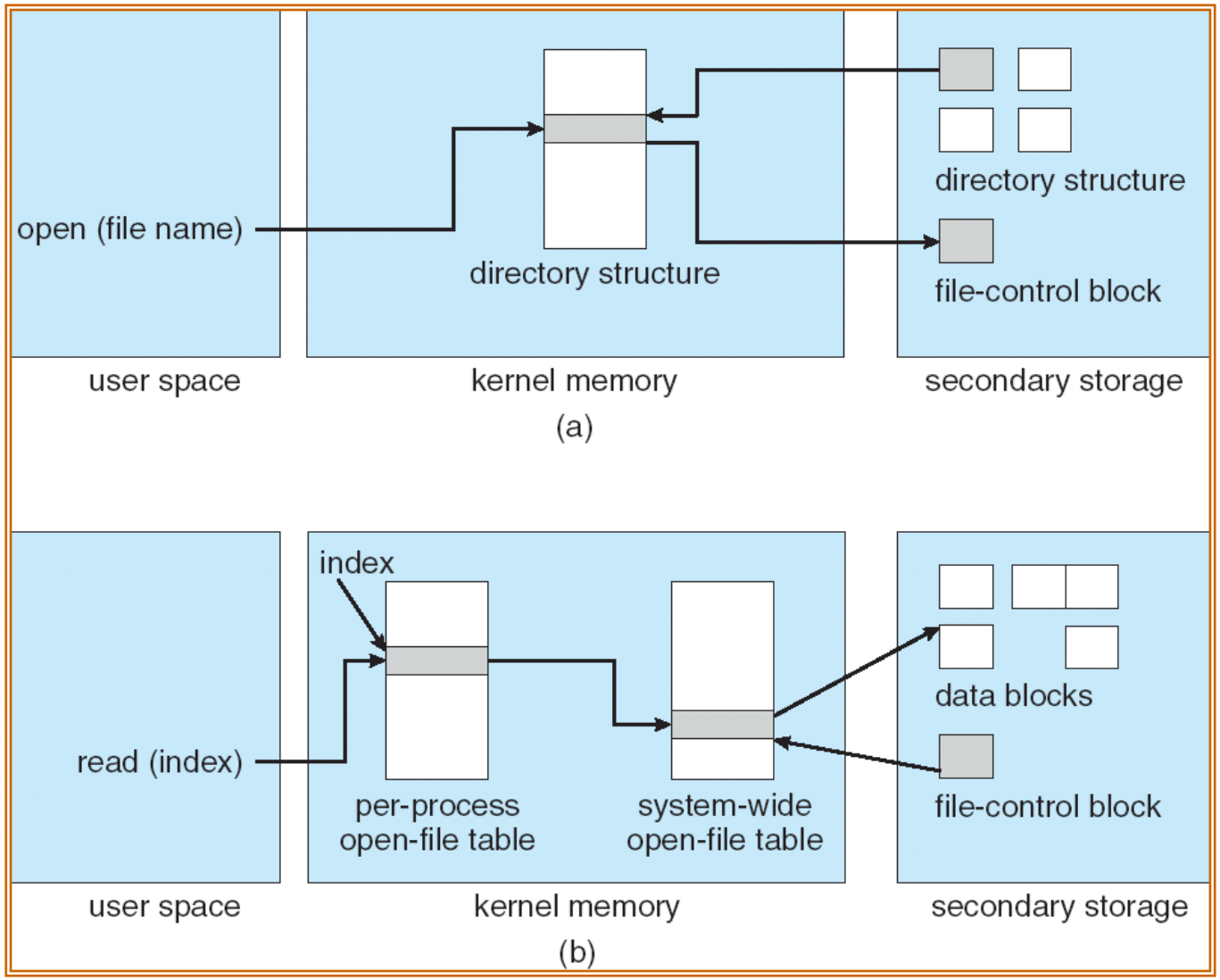


24

B. The software layer: the file abstraction

- File: memory abstraction used by the application and OS layers
  - linear array of bits/bytes
  - properties:
    - durable → information will not be changed in time
    - has a name →
    - allows access to individual bits/bytes → has a cursor which defines the current position in the file.
- The OS provides an API (Application Programming Interface) supporting a range of file manipulation operations.
- A user must first OPEN a file before accessing it and CLOSE it after it has finished with it. This strategy:
  - allows different access rights (READ, WRITE, READ-WRITE)
  - coordinate concurrent access to the file
- Some file systems
  - use OPEN and CLOSE to enforce before-or-after atomicity
  - support all-or-nothing atomicity → e.g., ensure that if the system crashes before a CLOSE either all or none of WRITEs are carried out

Open and Read operations



26