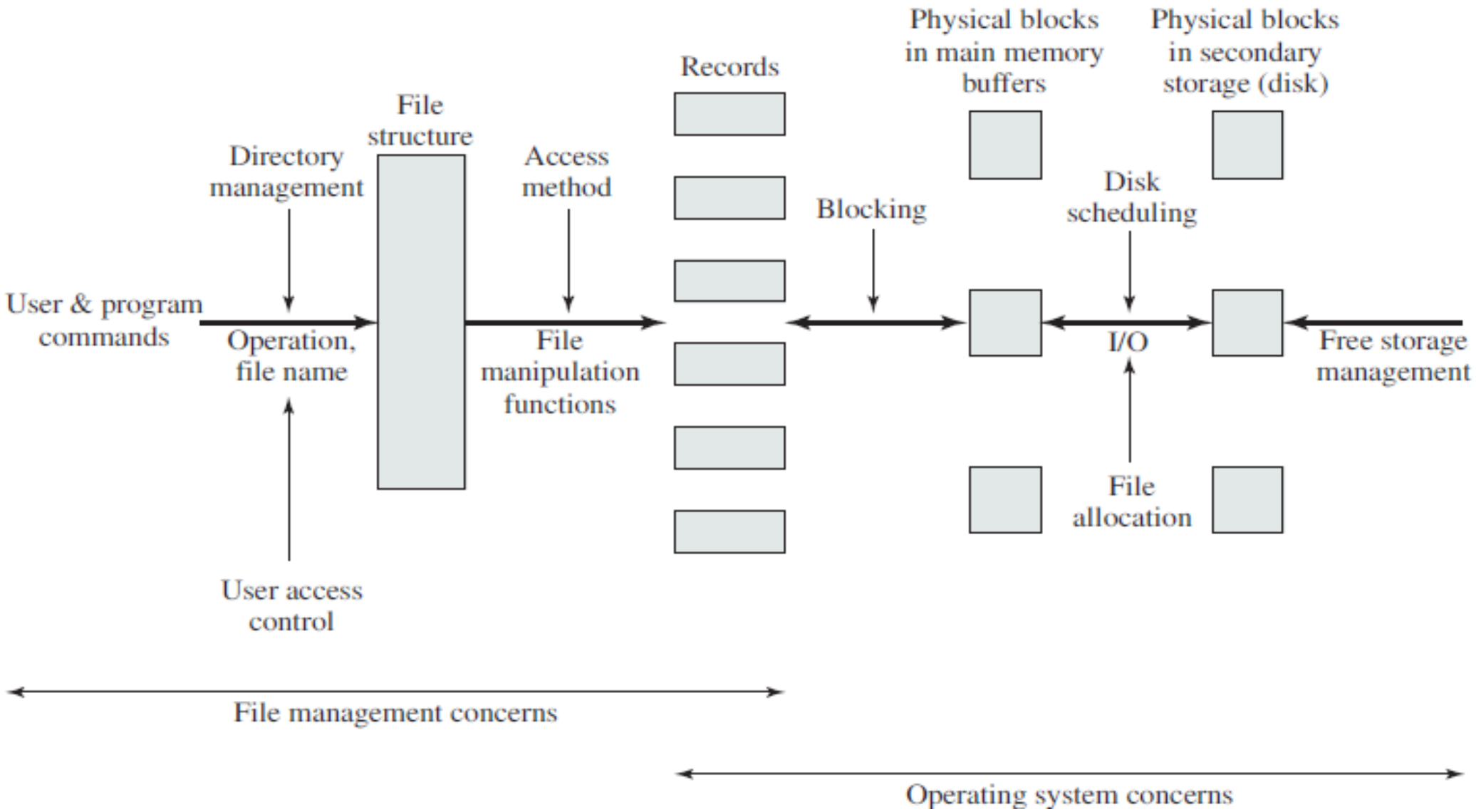
240-229: SDA (Operating Systems session)

Lecture 6: Storage & Device management

Associate Professor Dr. Sangsuree Vasupongayya
sangsuree.v@psu.ac.th
Department of Computer Engineering
Prince of Songkla University

File Management



File Concept

- Contiguous logical address space
- Types: Data, Program
- Structure
  - None - sequence of words, bytes
  - Simple record structure

    E.g., lines, Fixed length, Variable length
  - Complex Structures

    E.g., Formatted document
  - Who decides: Operating system, Program
- Operations E.g., Create, Read, Write, Delete
- Access method E.g., Direct, Sequential

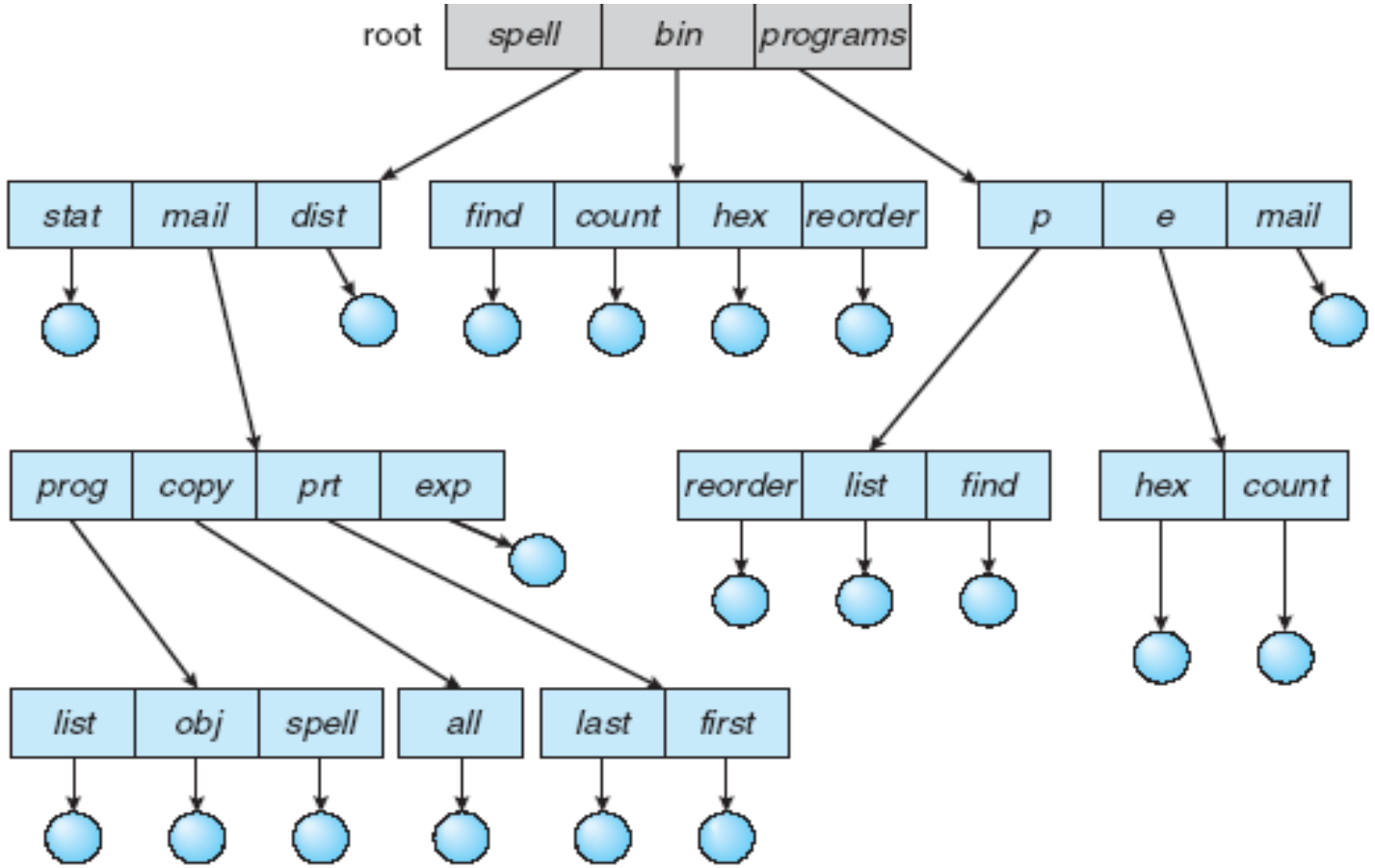
File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk

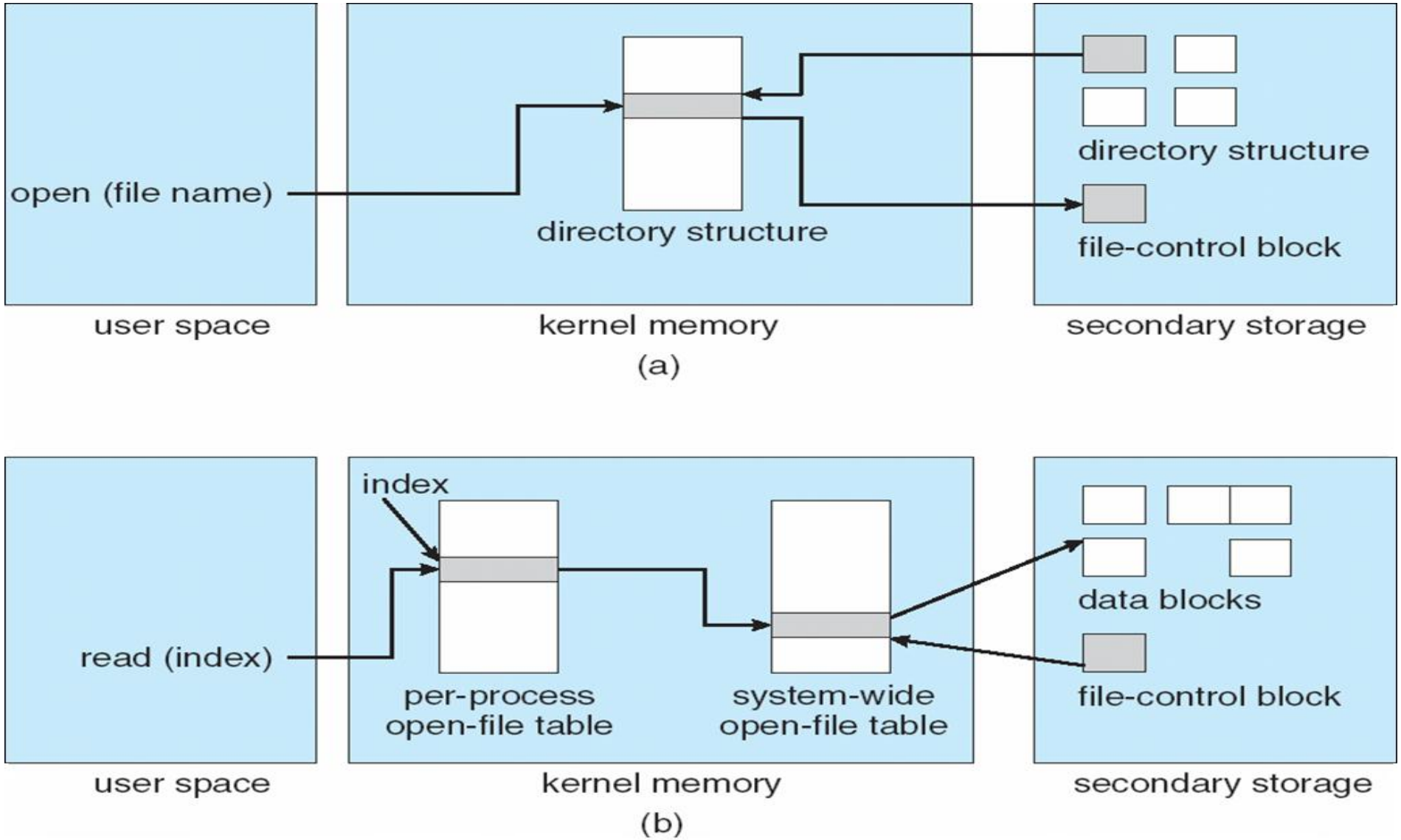
File system

- Organize the files
- Directory
  - A symbol table that translates file names into their directory entries
  - Operations

    E.g., Create, search, delete, list, rename, traverse
  - Various types
    - Simple directory
    - Two-level directory
    - Tree structure
    - Graph structure



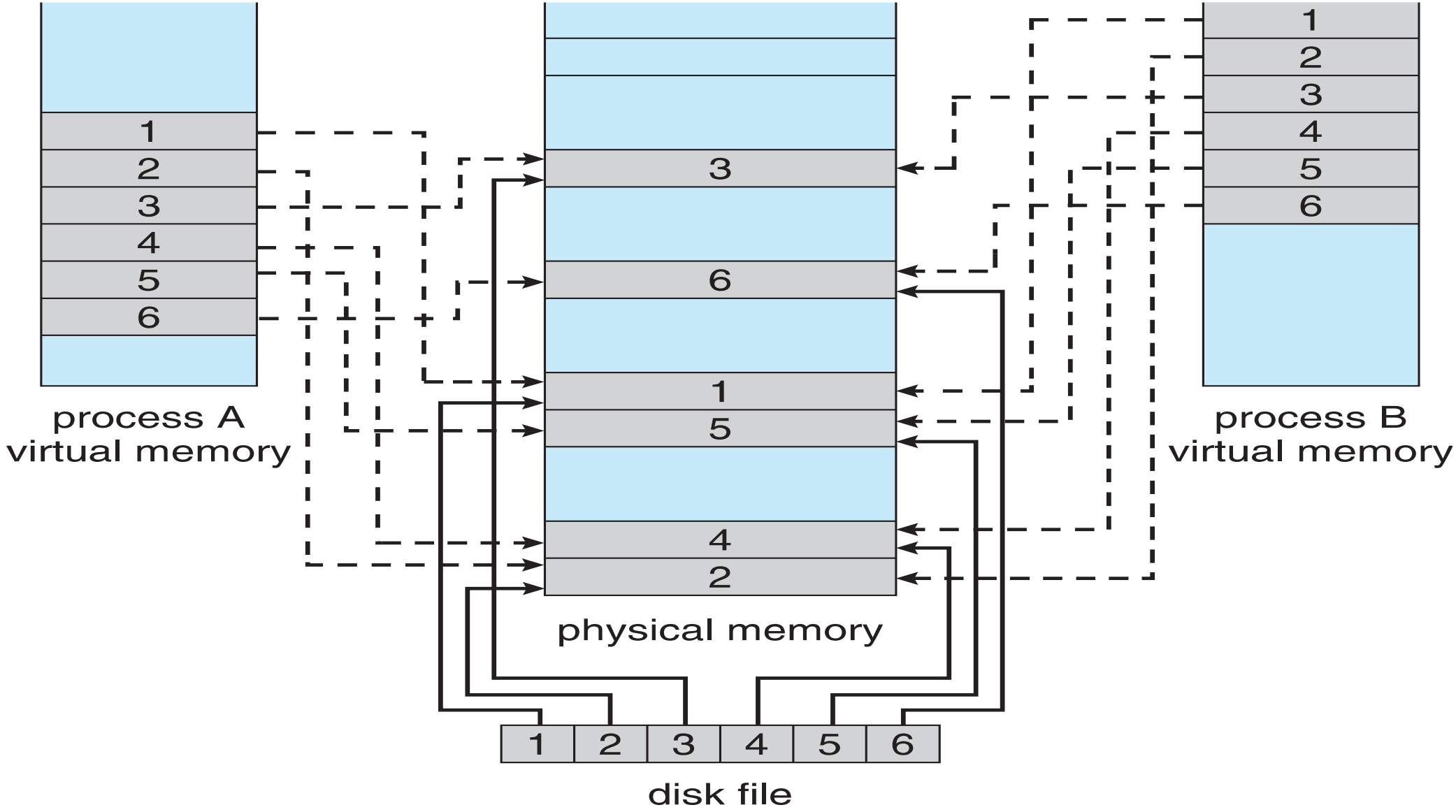
In-Memory File System Structures



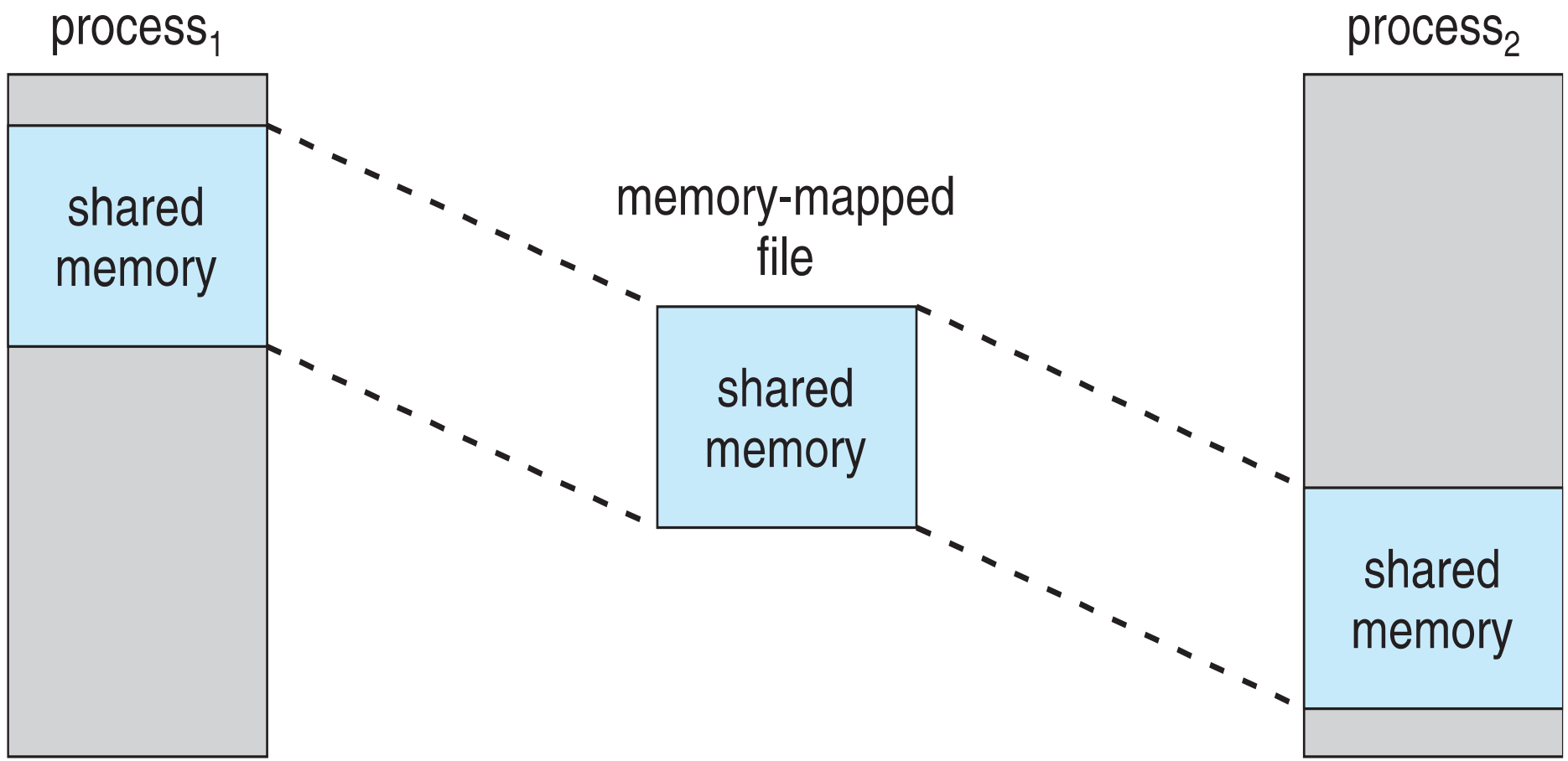
Memory-Mapped Files

- allows file I/O to be treated as routine memory access by **mapping** a disk block to a page in memory
- A file is initially read using demand paging
  - A page-sized portion of the file is read from the file system into a physical page
  - Subsequent reads/writes to/from the file are treated as ordinary memory accesses
- Simplifies and speeds file access
  - by driving file I/O through memory rather than `read()` and `write()` system calls
- Also allows several processes to map the same file allowing the pages in memory to be shared
- when does written data make it to disk?
  - Periodically and / or at file `close()` time
  - For example, when the pager scans for dirty pages

Memory Mapped Files



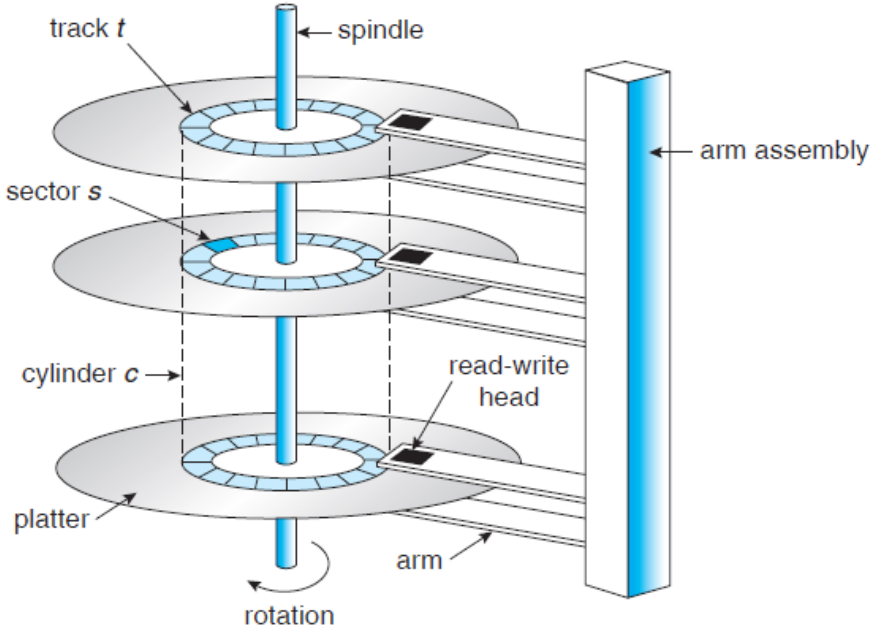
Shared Memory via Memory-Mapped I/O



The file can now be shared among processes

Storage

Magnetic disks

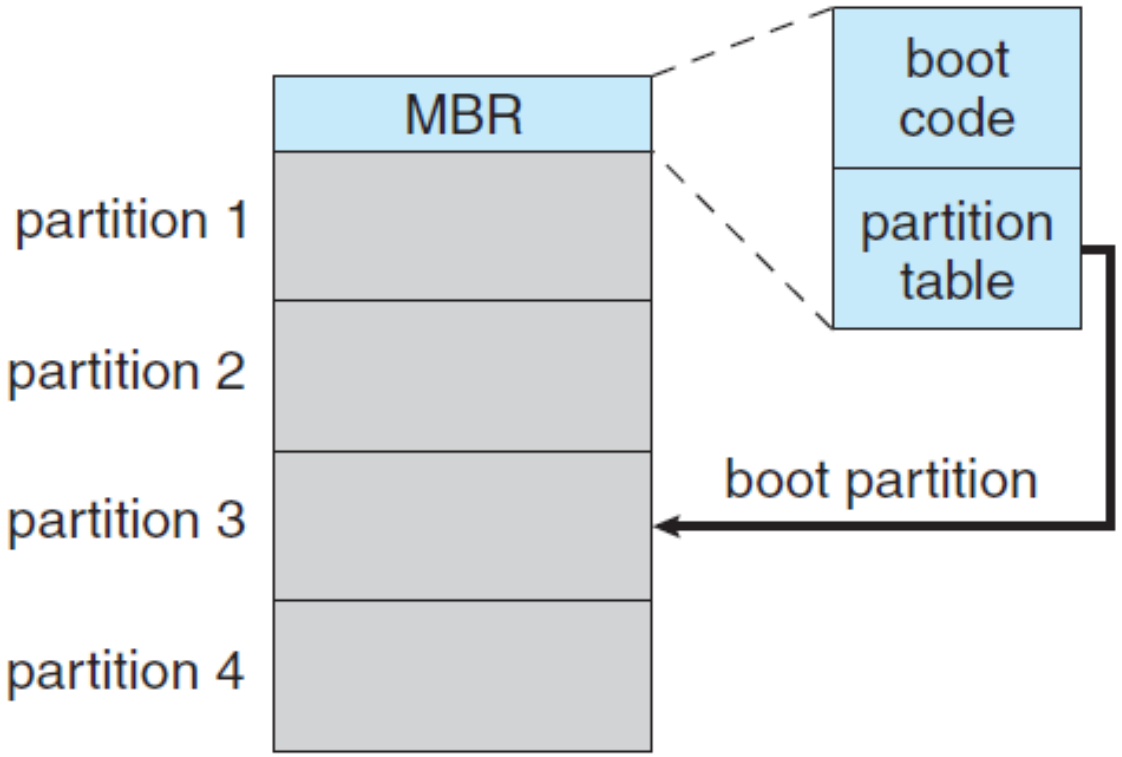


- Provide bulk of secondary storage
- Attached to a computer by a set of wires (I/O bus) E.g., ATA, SATA, eSATA, USB, FC
- A disk controller is built into each disk drive, Usually comes with a build-in cache

Solid-state disk (SSD)

- More reliable than HDDs
  - no moving parts
- Faster than HDDs
  - no seek time or rotational latency
- More expensive
- Less capacity

Booting from a storage device

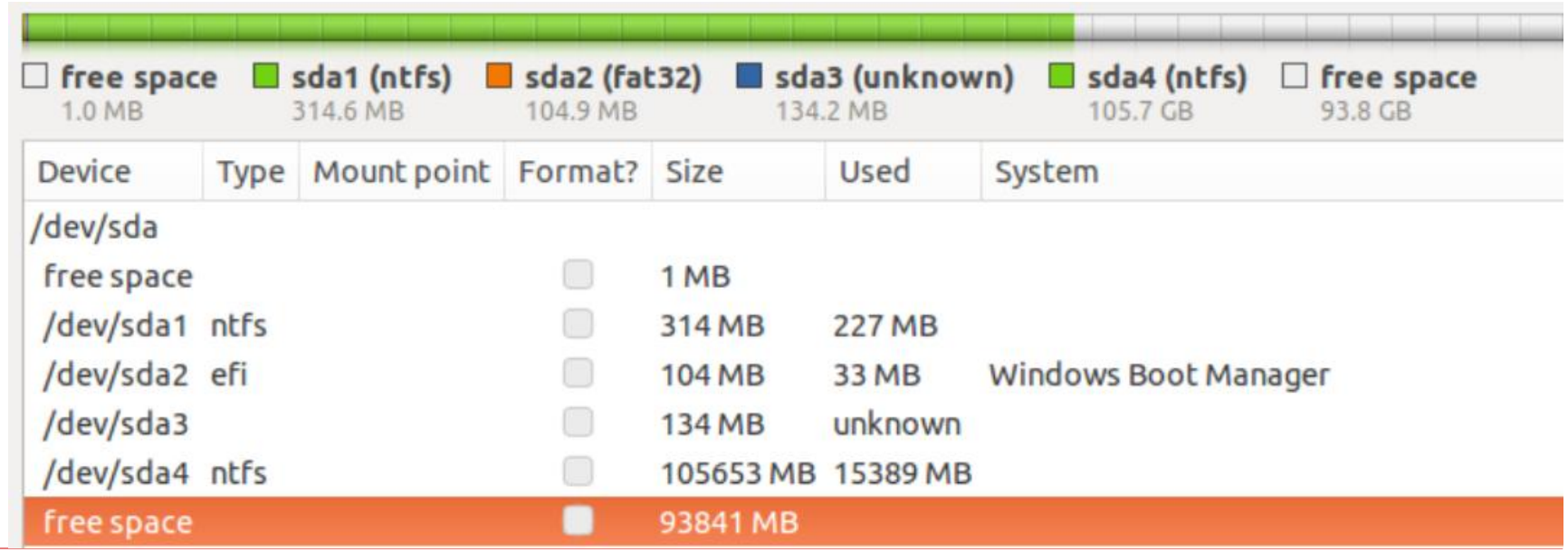


- MBR
  - Boot code
  - Partition table
- Boot partition
  - First sector or page will be read
  - Subsystems are then read

- Bootstrap reads the boot code from MBR
  - Understand storage controller & devices

Partitions

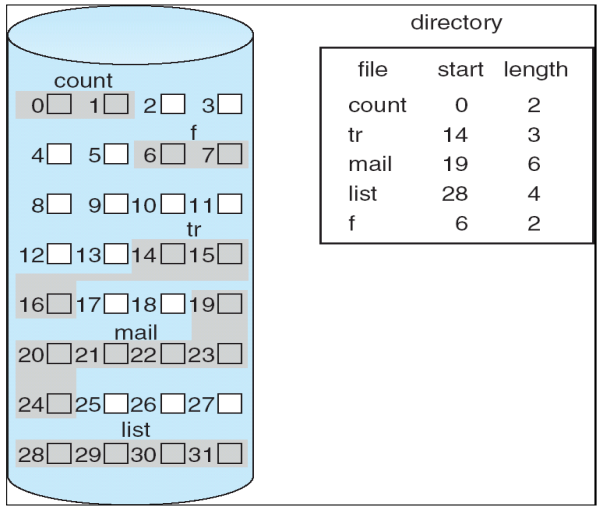
- A disk can be sliced into multiple partitions
- Partition can be “raw” (no file system) or “cooked”
- Boot information in a separate partition
  - At boot, the system does not have file-system device driver
  - Usually, the boot information is a sequential series of blocks, loaded as an image into memory
  - Dual-booted: multiple operating systems can be installed on such a system

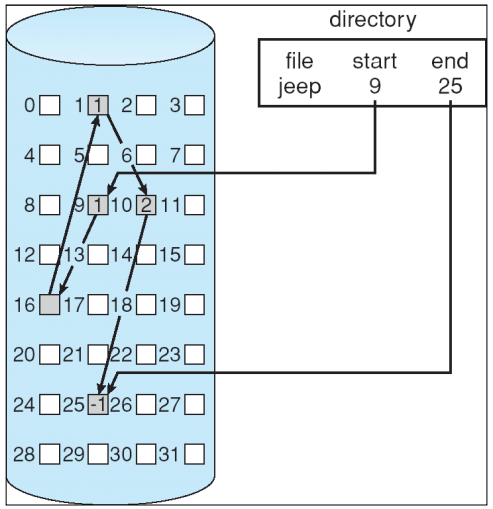


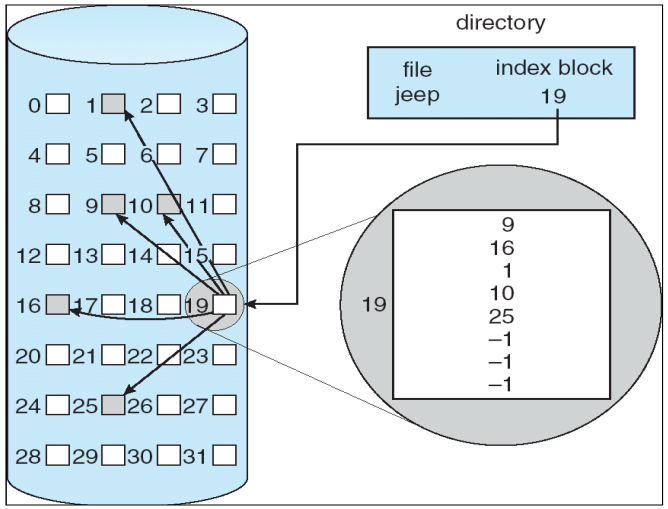
Device	Type	Mount point	Format?	Size	Used	System
/dev/sda						
free space				1 MB		
/dev/sda1	ntfs			314 MB	227 MB	
/dev/sda2	efi			104 MB	33 MB	Windows Boot Manager
/dev/sda3				134 MB	unknown	
/dev/sda4	ntfs			105653 MB	15389 MB	
free space				93841 MB		

Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
  - **Contiguous allocation**
  - **Linked allocation**
  - **Indexed allocation**

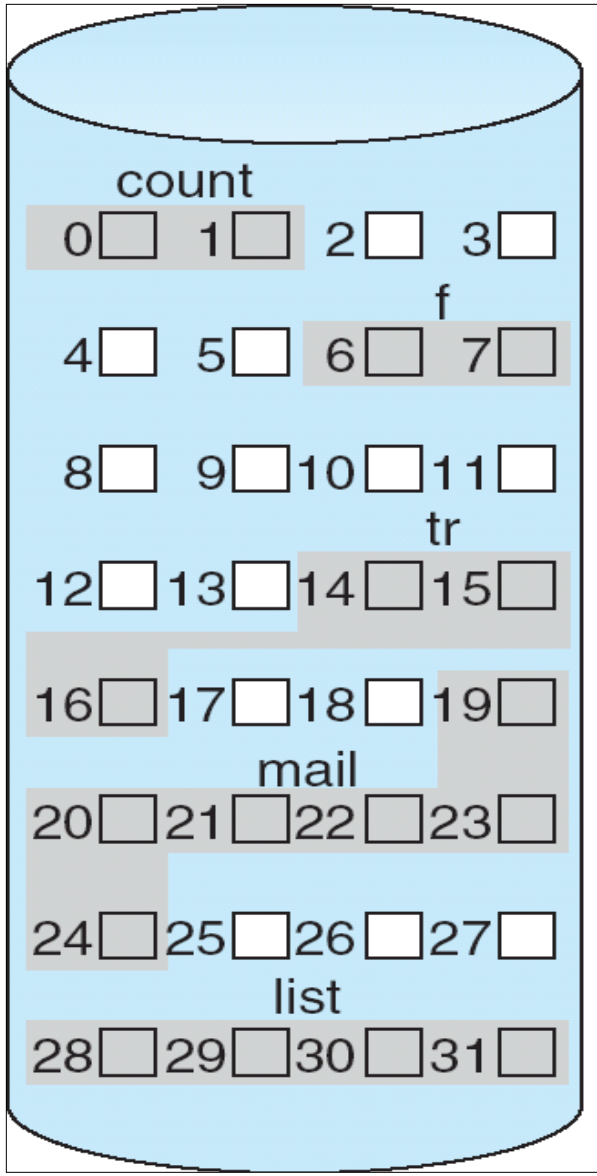






Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Random access
- Wasteful of space (dynamic storage-allocation problem)
- Files cannot grow (solution: Extent)
  - An extent is a contiguous block of disks
  - A file consists of one or more extents.

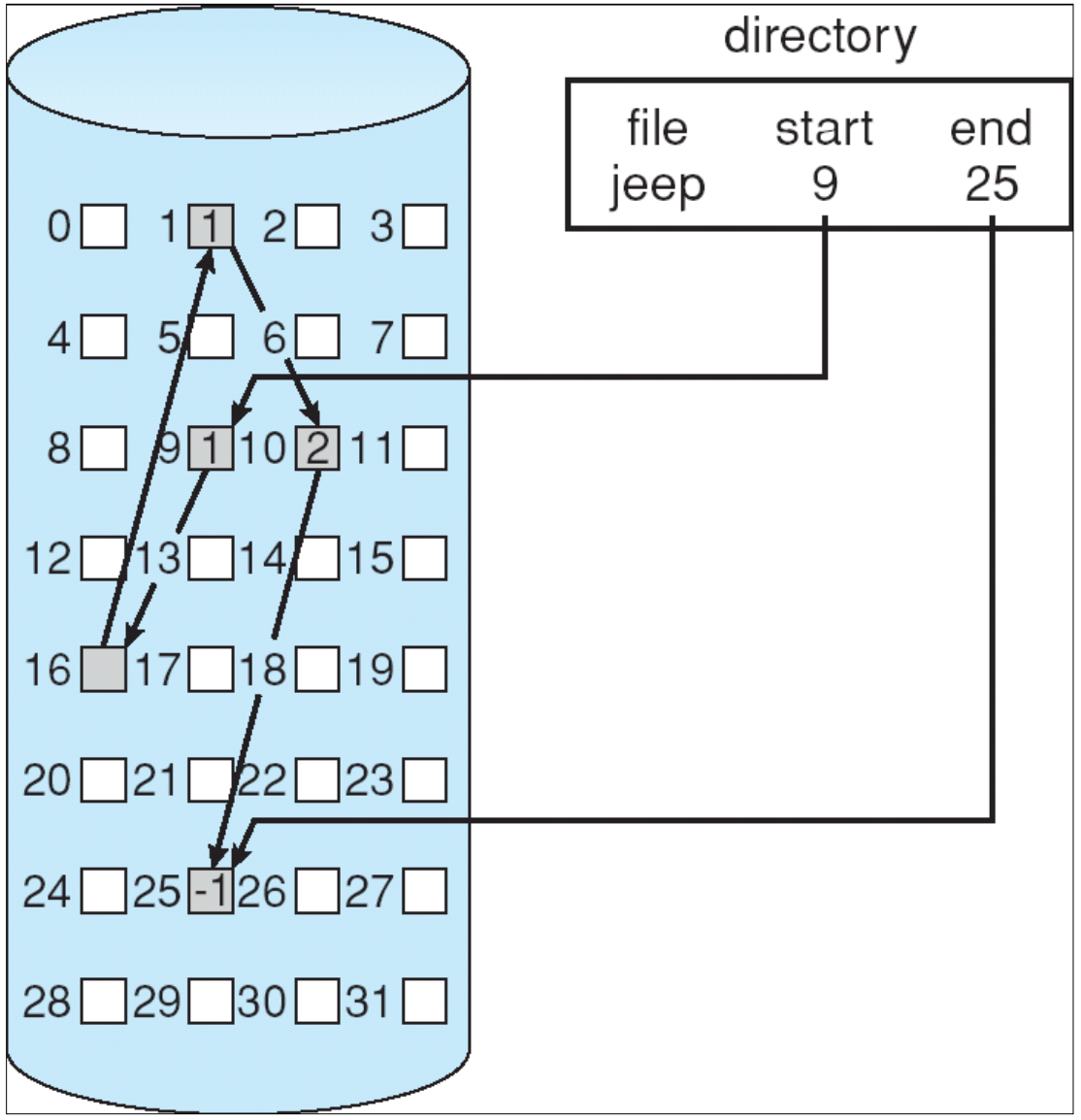


directory

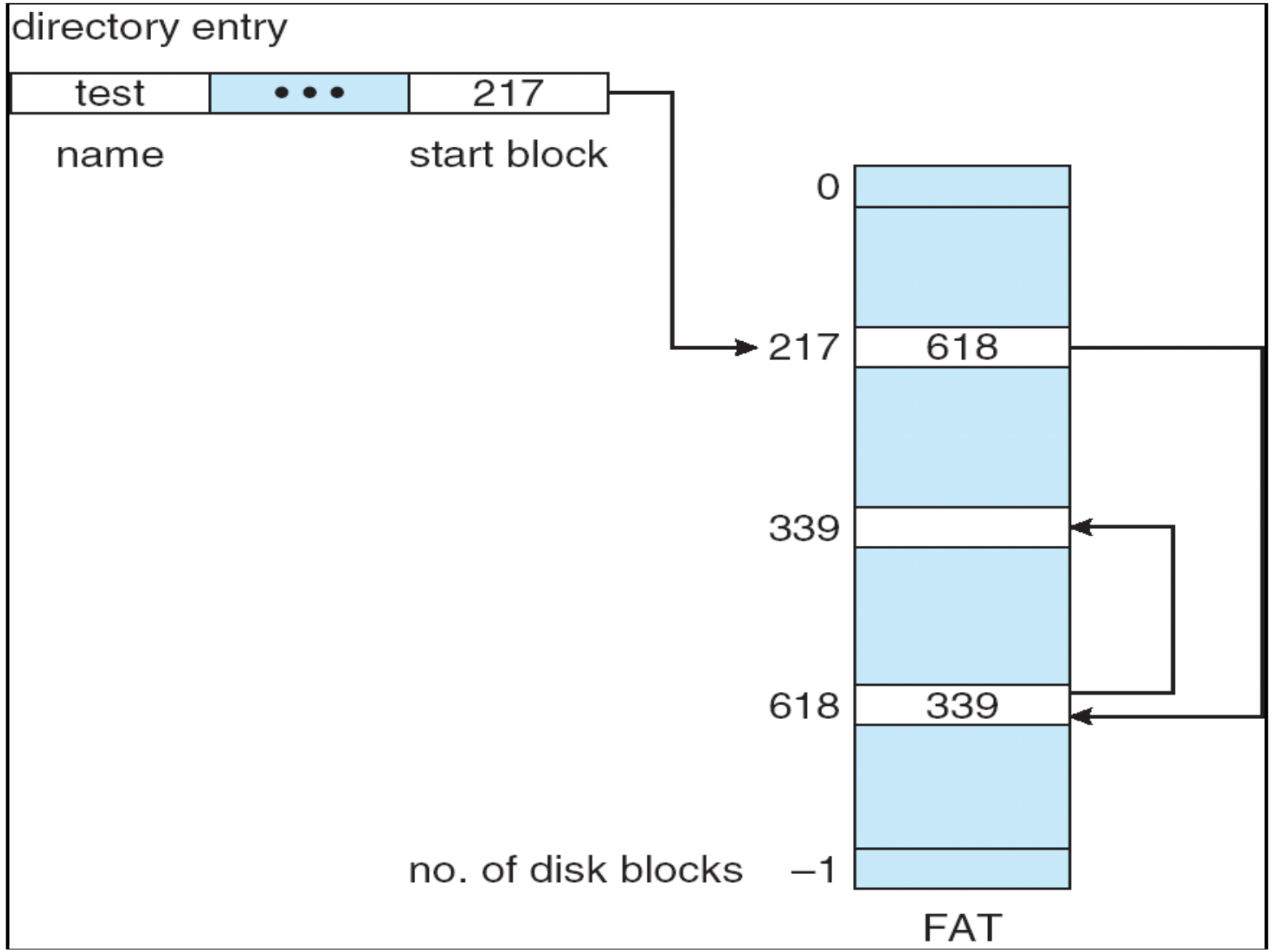
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked Allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
- Simple – need only starting address
- Free-space management system – no waste of space
- No random access

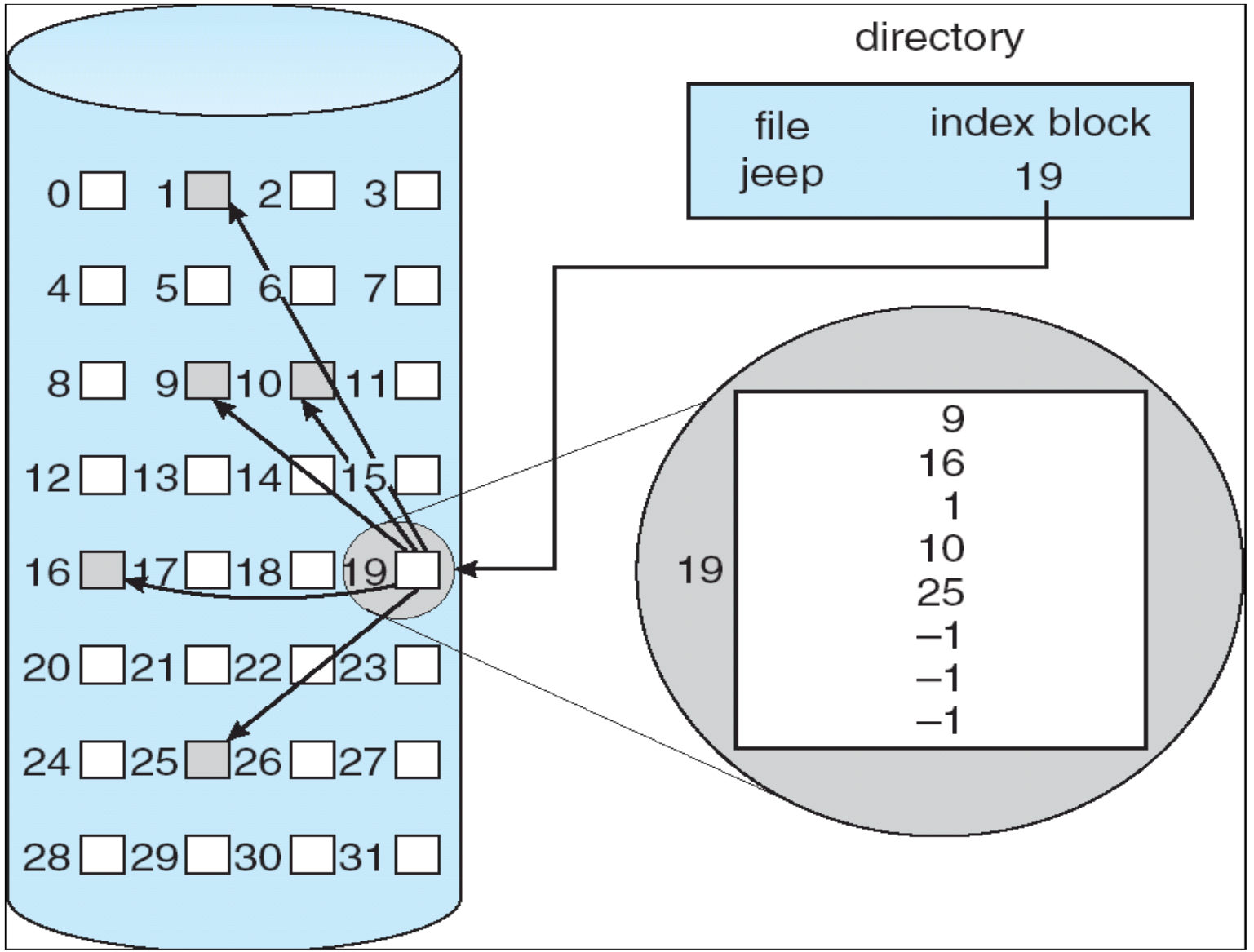


Example: File-Allocation Table (FAT)



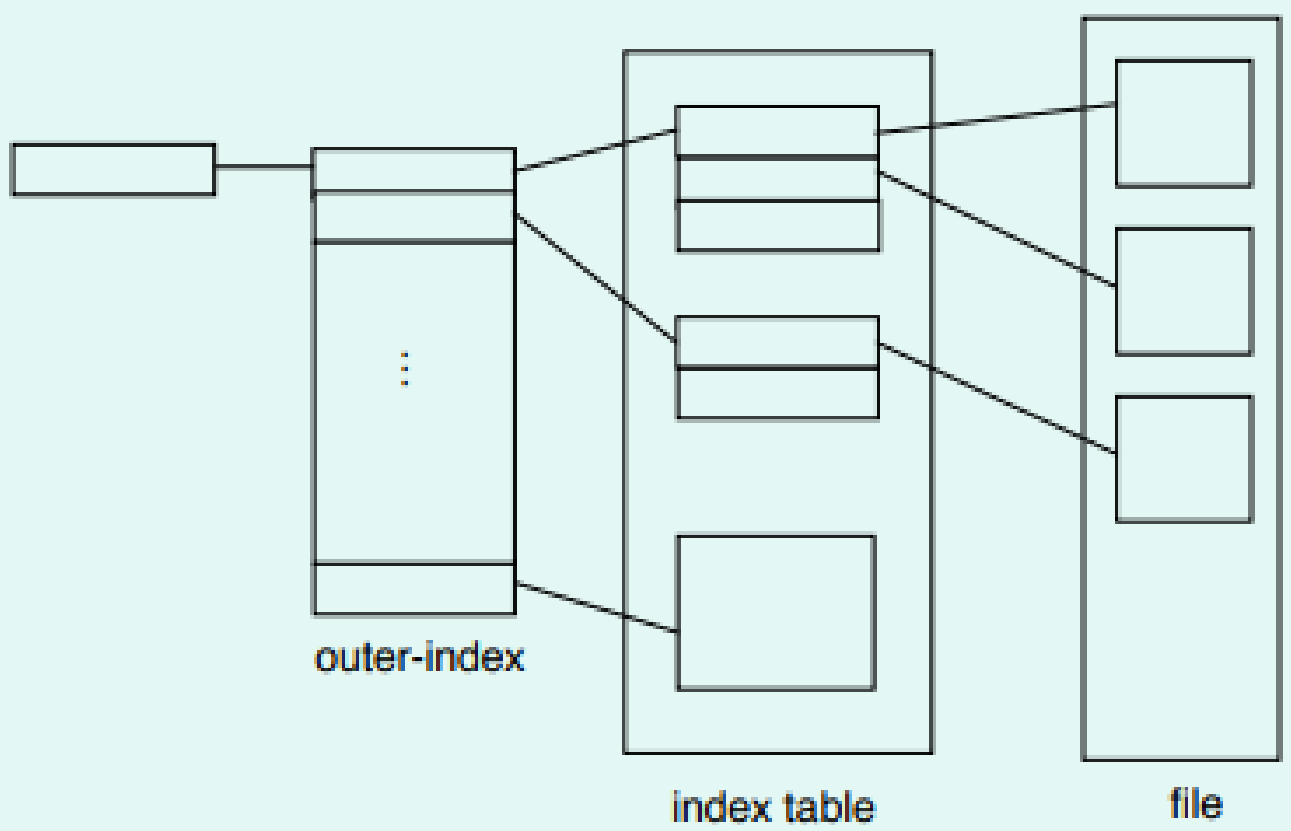
Indexed Allocation

- Brings all pointers together into the *index block.*
- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.

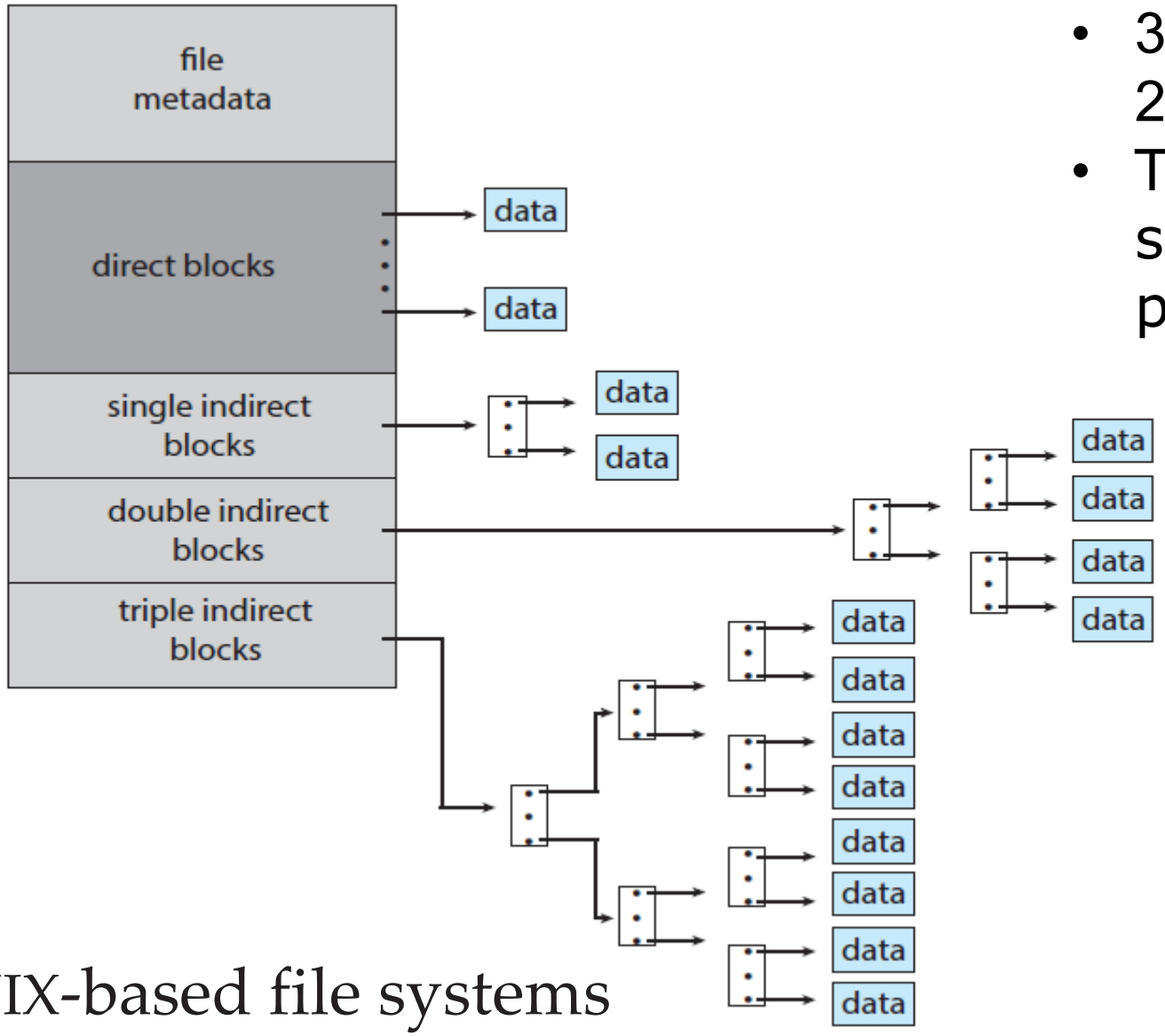


Index allocation implementation

- Linked scheme
  - An index block is normally one storage block
  - For large files, several index blocks are linked
- Multilevel index
  - First level points to a block which points to data



Combine scheme

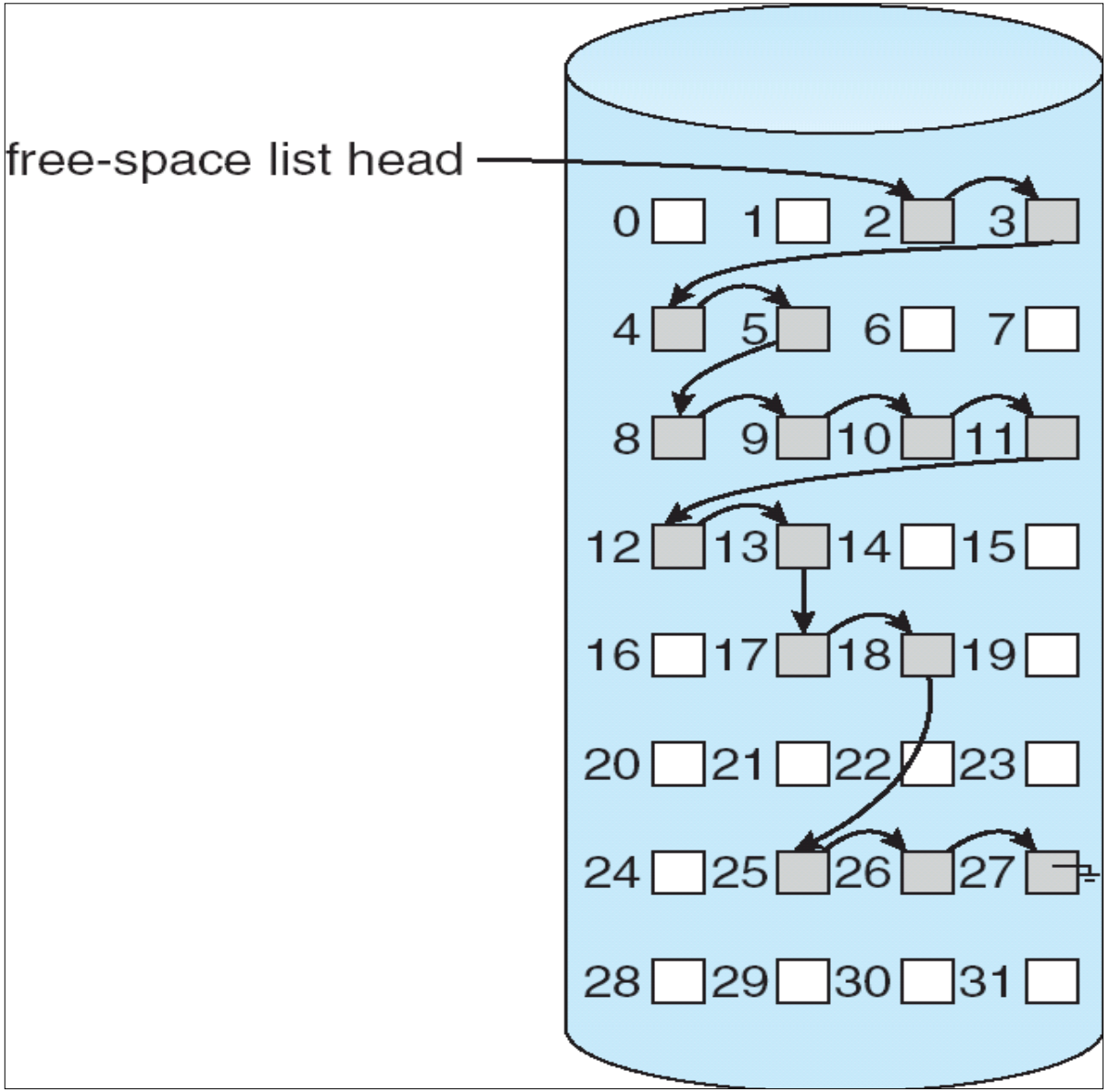


- 32-bit file pointer reaches only 2^{32} bytes, or 4 GB
- The ZFS file system supports 128-bit file pointers

UNIX-based file systems

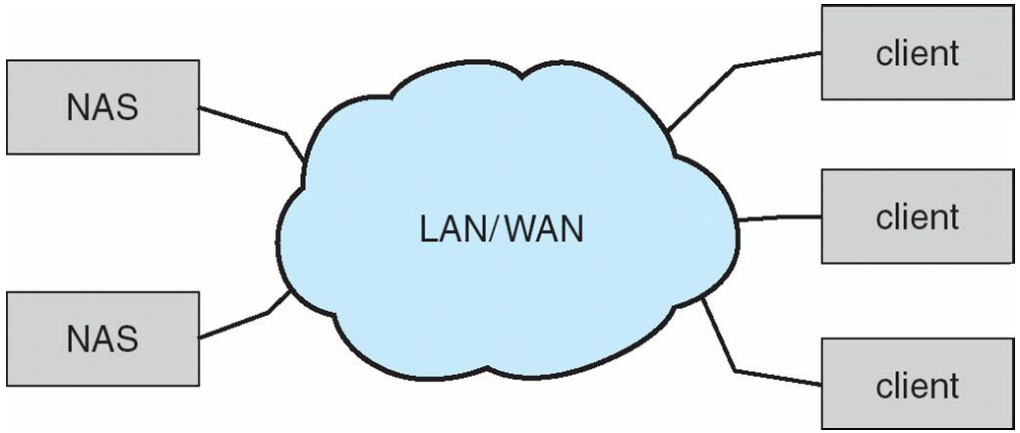
Free-Space Management

- Keep a free-space list
- Implementation
  - Bit vector
  - Linked list
  - Grouping
  - Counting



SAN and NAS

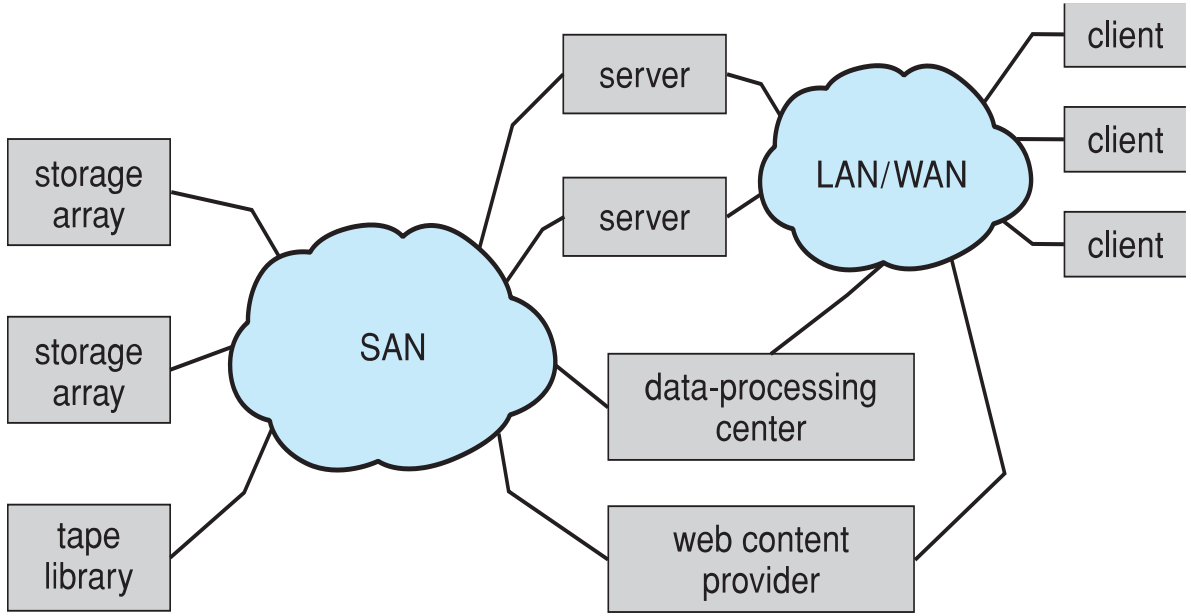
Network-attached storage (NAS)



NAS is storage made available over a network rather than over a local connection (such as a bus), Remotely attaching to file systems

Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network

Storage Area Network (SAN)



Common in large storage environments
Multiple hosts attached to multiple storage arrays - flexible

I/O Hardware

- Incredible variety of I/O devices
- 3 categories
  - Human readable
  - Machine readable
  - Communications
- Common concepts
  - Port
  - Bus (daisy chain or shared direct access)
  - Controller (host adapter)

I/O Port

- Typically consists of 4 registers
  - Data-in : read in by the host to get input
  - Data-out : written by the host to send output
  - Status: contains bits that indicate states of the device
  - Control: can be written by the host to start a command or to change the mode of a device
- Some controllers have a buffer to expand the size of the data registers. Thus, a small burst of data can be hold until the device or the host is able to receive the data.

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including
    - buffering : storing data temporarily while it is being transferred
    - caching : storing parts of data in faster storage for performance
    - spooling : the overlapping of output of one job with input of other jobs
  - General device-driver interface
  - Drivers for specific hardware devices

Polling

Determines state of device

- command-ready
- busy
- Error

Busy-wait cycle to wait for I/O from device

When to use?

- If the controller and device are fast, this method is a reasonable one
- If the host must service the device quickly, or data will be lost.

Interrupts

CPU **Interrupt-request line** triggered by I/O device

Interrupt handler receives interrupts

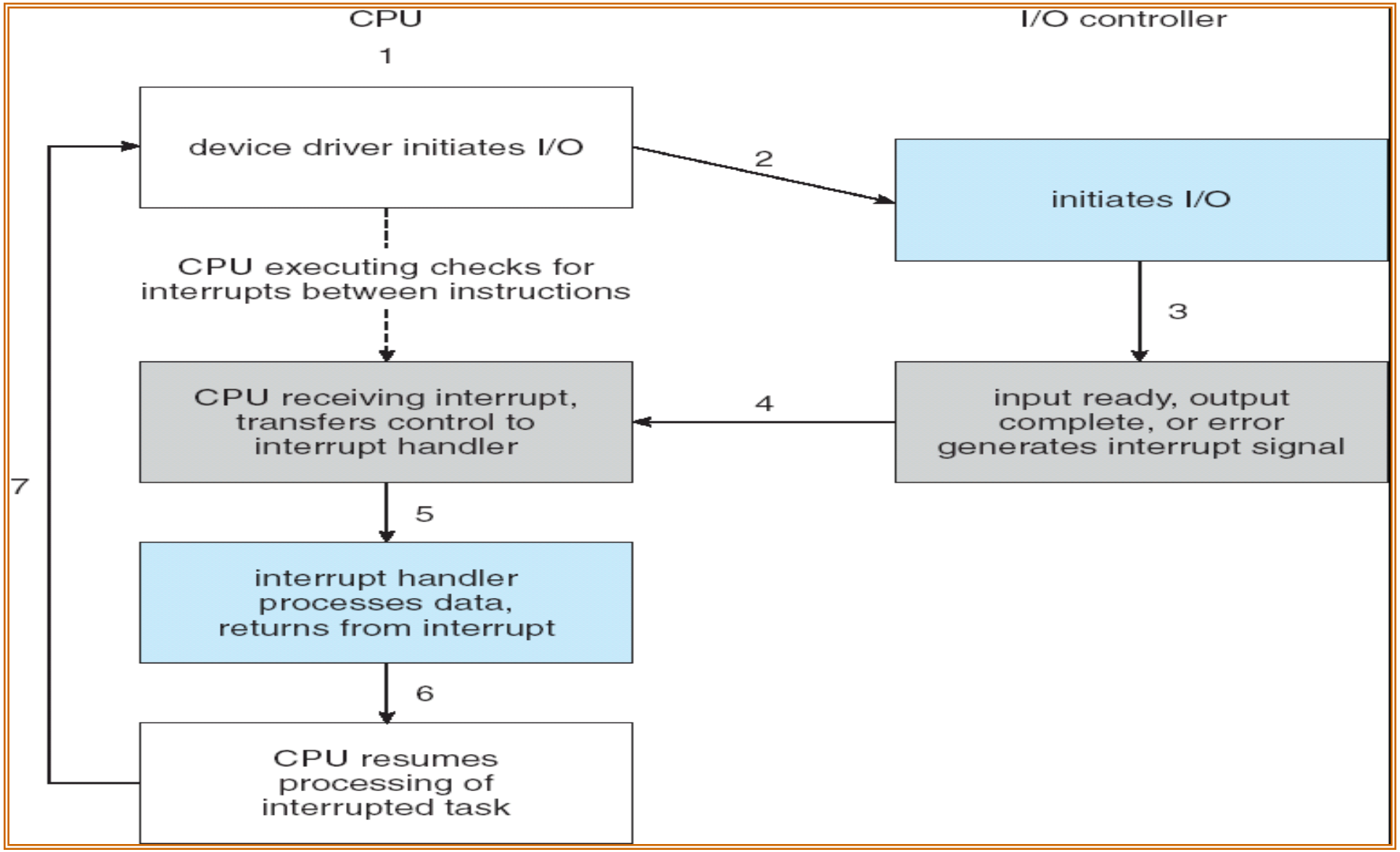
Maskable to ignore or delay some interrupts

Interrupt vector to dispatch interrupt to correct handler

- Based on priority
- Some **nonmaskable**

Interrupt mechanism also used for exceptions

Interrupt-Driven I/O Cycle

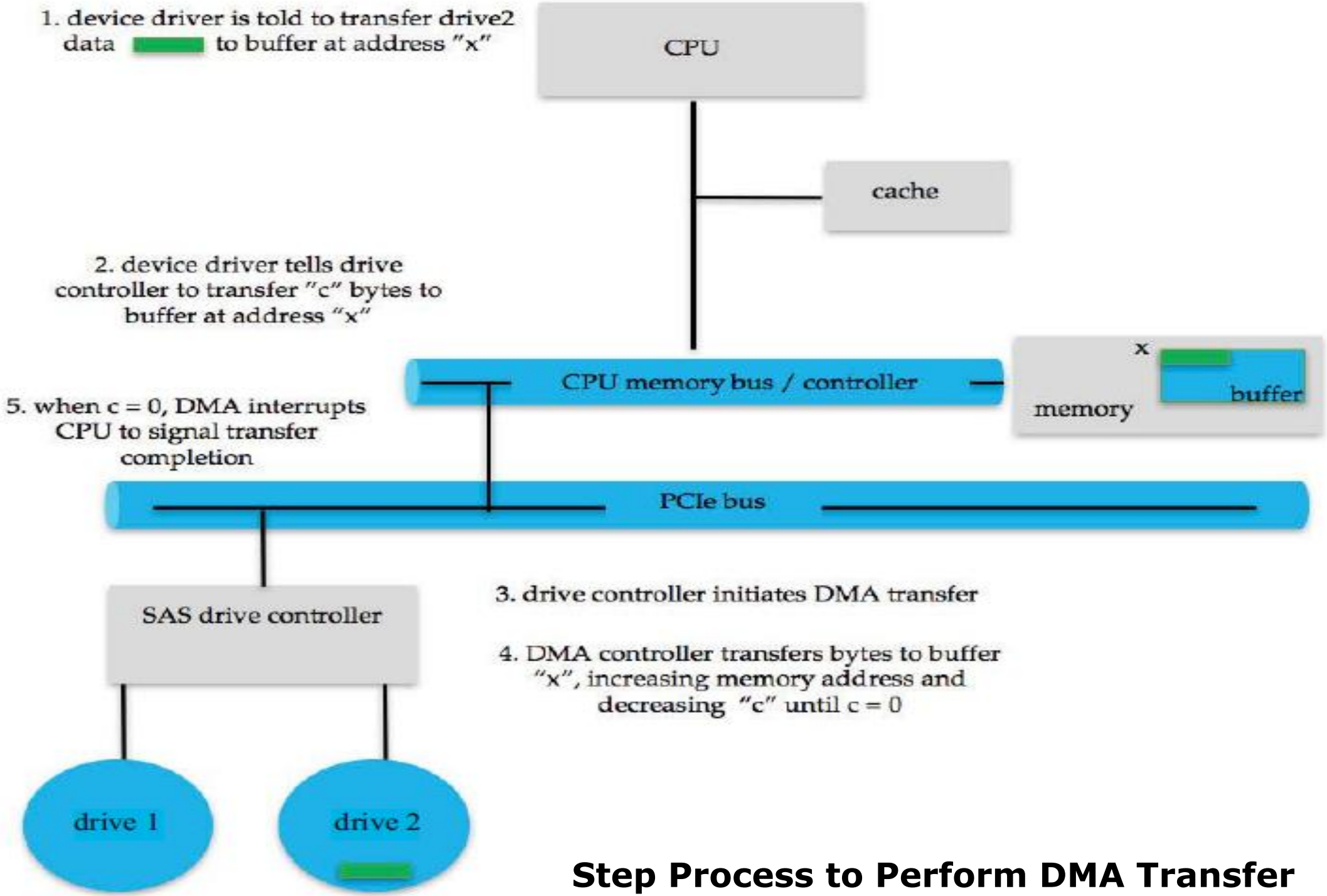


Direct Memory Access

The interrupt-driven I/O is good for small amounts of data.

- Used to avoid **programmed I/O** for large data movement
- Requires **DMA** controller
- Bypasses CPU to transfer data directly between I/O device and memory

- Some computer architectures use physical memory addresses for DMA, but others perform Direct virtual memory (DVMA) using virtual addresses that undergo translation to physical addresses. DVMA can perform a transfer between two memory-mapped devices without the intervention of the CPU or the use of main memory



Step Process to Perform DMA Transfer

Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential or random-access**
  - **Sharable or dedicated**
  - **Speed of operation**
  - **read-write, read only,** or **write only**

Application I/O interface

structuring techniques and interfaces for the operating system that enable I/O devices to be treated in a standard, uniform way

- Abstraction
  - abstract away the detailed differences in I/O devices by identifying a few general kinds -- **Interface**
- Encapsulation
  - The differences are encapsulated in kernel modules called **device drivers**
- Software layering
  - I/O-related portions of the kernel are structured in software layers

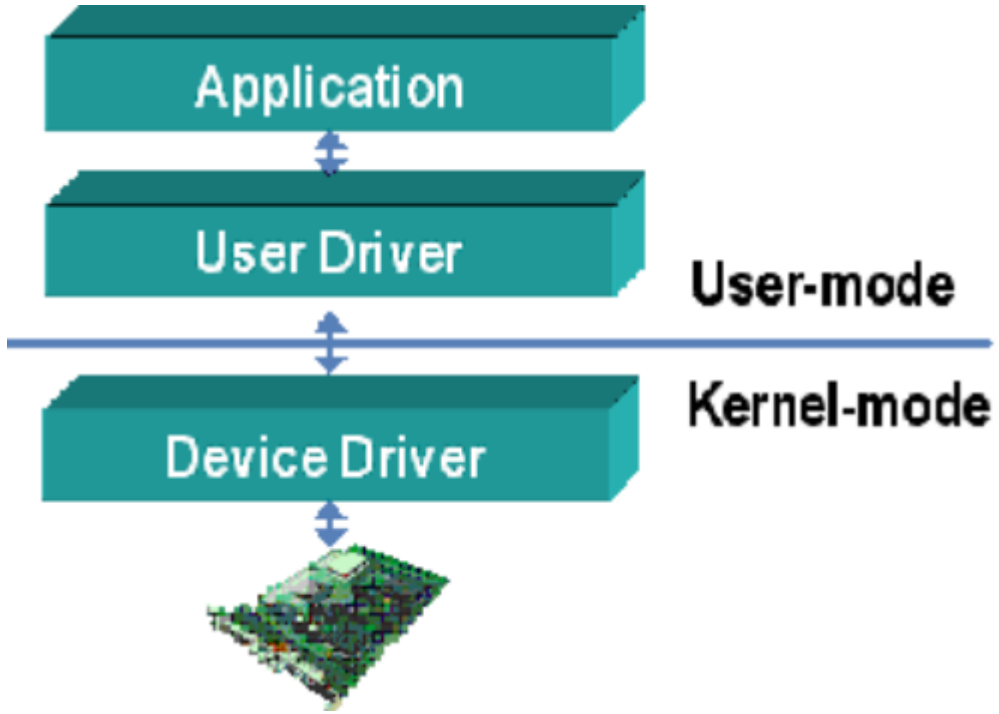
Device Driver

Purpose

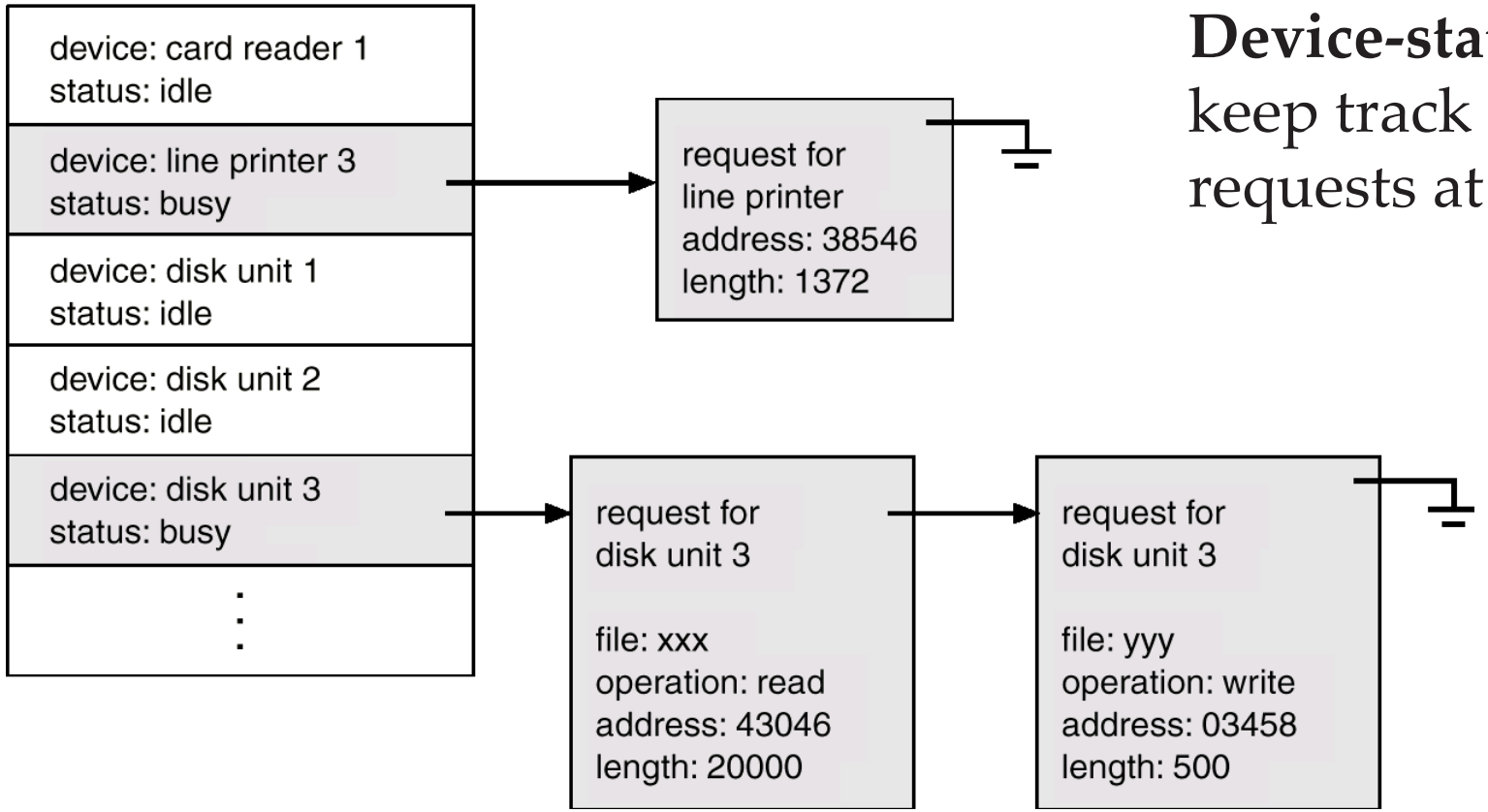
- Understands the device controller
- Presents a uniform interface to the device

During I/O operation (Read)

- Device driver: loads the appropriate registers within the device controller
- Device controller:
  - examines the contents of the registers to determine the actions
  - transfer data from device to local buffer
  - tells the device driver once it is done
- Device driver: interrupt the CPU



Device-Status Table



Device-status table to keep track of many I/O requests at the same time

the operating system might attach the wait queue to a **device-status table**, which contains an entry for each I/O device

Blocking and Nonblocking I/O

Blocking - process suspended until I/O completed

- Easy to use and understand
- Insufficient for some needs

Nonblocking - I/O call returns as much as available

- User interface, data copy (buffered I/O)
- Implemented via multi-threading
- Returns quickly with count of bytes read or written

Asynchronous - process runs while I/O executes

- Not block
- I/O subsystem signals process when I/O completed

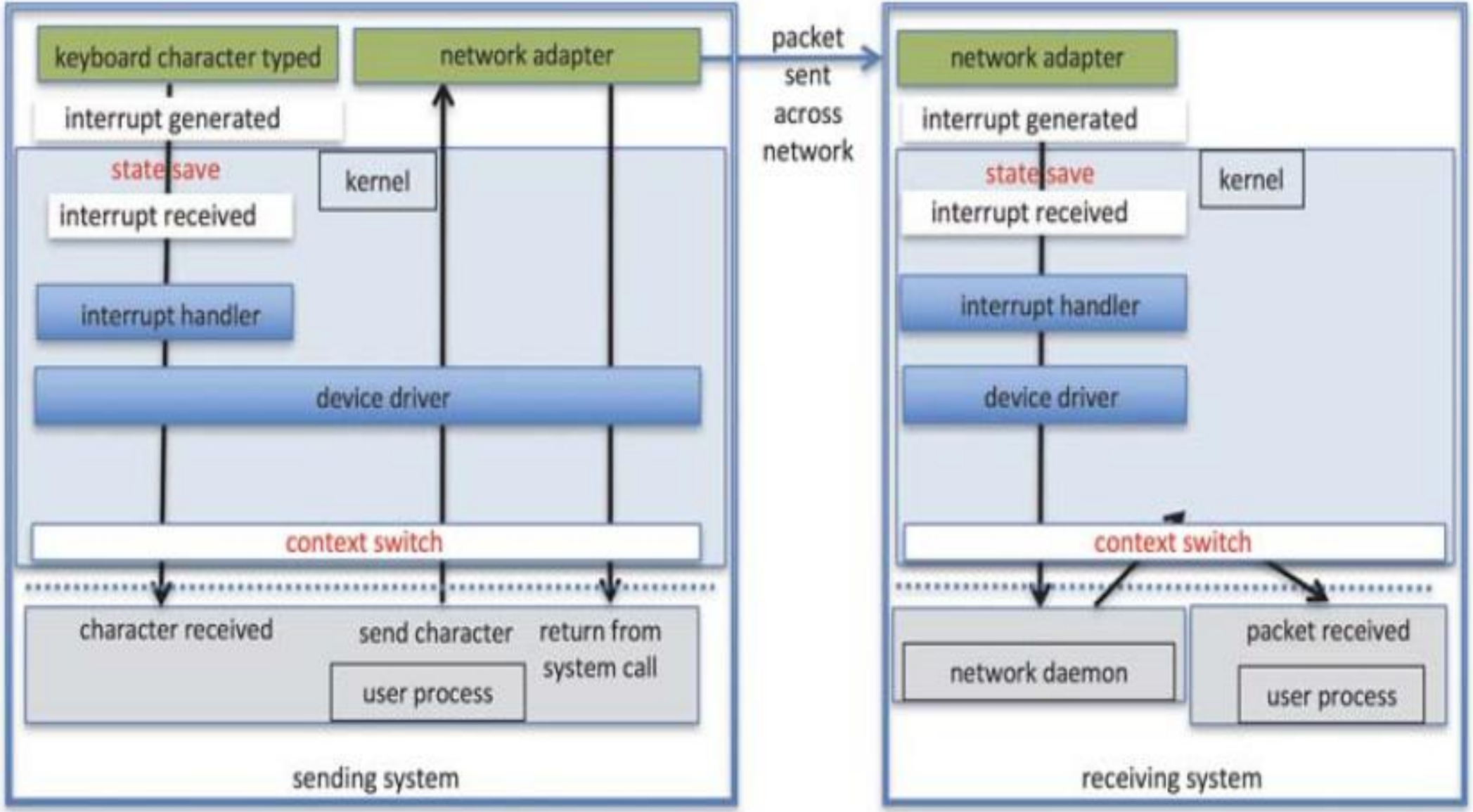
A keyboard example

- Character-stream interface
  - basic system calls of this interface is to get() or put() one character
  - higher, libraries can be built that offer line-at-a-time access, with buffering and editing services
  - E.g., when a user types a backspace, the preceding character is removed from the input stream
- This style of access is convenient for input devices such as keyboards, mice, and modems that produce data for input "spontaneously" —that is, at times that cannot necessarily be predicted by the application.

Network Devices

- One interface available in many operating systems, including UNIX and Windows, is the network **Socket** interface.
- Varying enough from block and character to have own interface
- Unix and Windows NT/9*x*/2000 include socket interface
  - Separates network protocol from network operation
  - Includes `select` functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

Intercomputer communication



Clocks and Timers

- provide three basic functions:
  - Give the current time
  - Give the elapsed time
  - Set a timer to trigger operation *X* at time *T*
- **Programmable interval timer** used for timings, periodic interrupts
- Modern PCs include a **High-performance event timer (HPET)** which runs at rates in the 10-megahertz range
- NTP, the **Network-time protocol**, which uses sophisticated latency calculations to keep a computer's clock accurate almost to atomic-clock levels

Kernel I/O Subsystem

- Scheduling: ordering via per-device queue, fairness
- Buffering - store data in memory while transferring between devices
  - device speed or transfer size mismatch
- Caching - fast memory holding copy of data, a copy for performance
- Spooling - hold output for a device
  - device can serve only one request at a time e.g., Printing
- Device reservation - provides exclusive access to a device
  - System calls for allocation, Watch out for deadlock
- Error Handling
  - OS can recover from disk read, device unavailable, transient write failures
  - Most return an error number or code when I/O request fails
  - System error logs hold problem reports
- I/O Protection
  - Performed via system call, I/O instructions must be privilege

Power Management

- Heat generated by device, require cooling
- First OS aspect in mobile computing
  - Component-level power management
    - Understands relationship between components
    - Turn unused component off
  - Wake locks
    - Prevent sleep of device
  - Power collapse
    - Marginal power use
    - Only awake enough to response (button press, incoming call)