BUCKNELL UNIVERSITY
Computer Science

CSCI 315 Operating Systems Design

I/O Systems

Notice: The slides for this lecture have been largely based on those accompanying an earlier edition of the course text *Operating Systems Concepts with Java*, by Silberschatz, Galvin, and Gagne. Many, if not all, of the illustrations contained in this presentation come from this source.

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.

Characteristics of I/O Devices

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only readÐwrite	CD-ROM graphics controller disk

Block and Character Devices

- Block devices include disk drives.
  - Commands include `read()`, `write()`, `seek()`.
  - Raw I/O or file-system access.
  - Memory-mapped file access possible.
- Character devices include keyboards, mice, serial ports.
  - Commands include `get()`, `put()`.
  - Libraries layered on top allow line editing.

Network Devices

- Different enough from block and character to have their 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).

Clocks and Timers

- Provide:
  - current time,
  - elapsed time,
  - timer.
- If programmable interval time used for timings, periodic interrupts.
- `ioctl` (on UNIX) covers odd aspects of I/O such as clocks and timers.

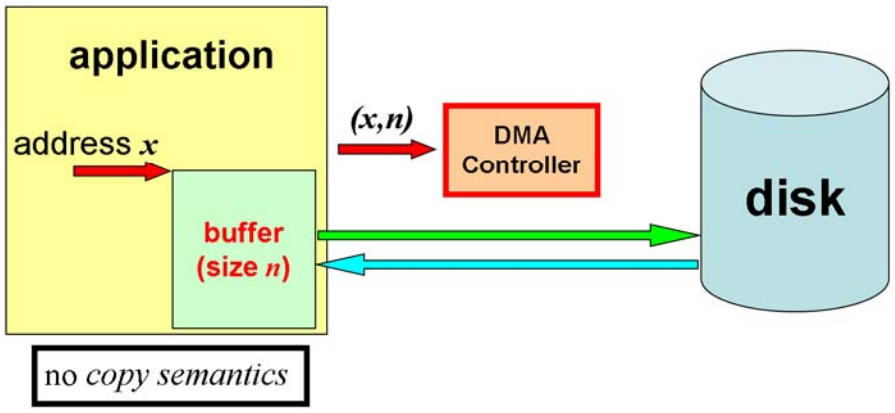
Blocking and Nonblocking I/O

- **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.
  - Difficult to use.
  - I/O subsystem signals process when I/O completed.

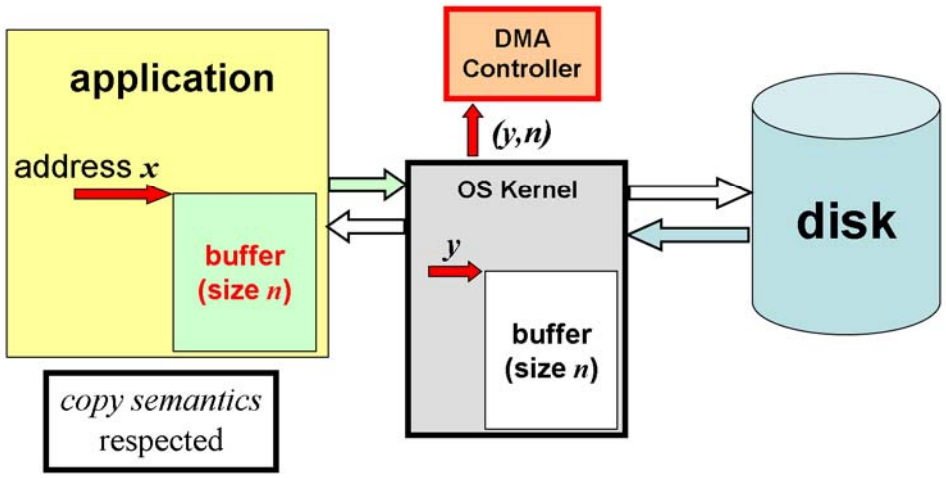
Kernel I/O Subsystem

- **Scheduling**
  - Some I/O request ordering via per-device queue.
  - Some OSs try fairness.
- **Buffering** - store data in memory while transferring between devices:
  - To cope with device speed mismatch.
  - To cope with device transfer size mismatch.
  - To maintain “copy semantics”.

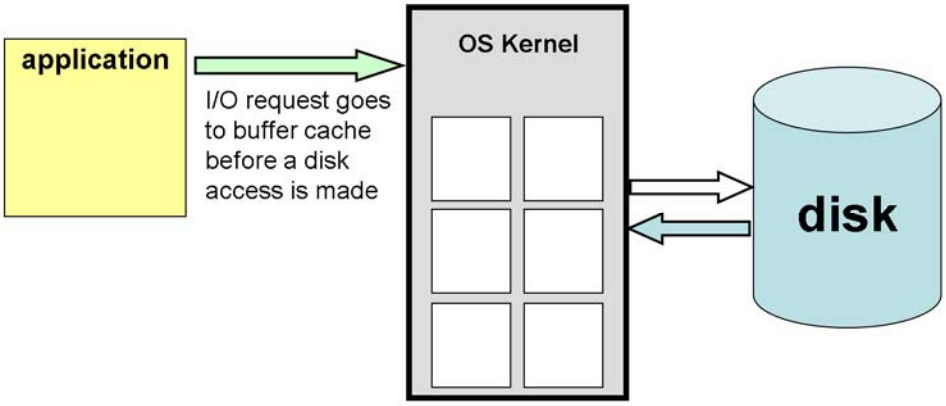
No Buffering



Buffering in Kernel Space

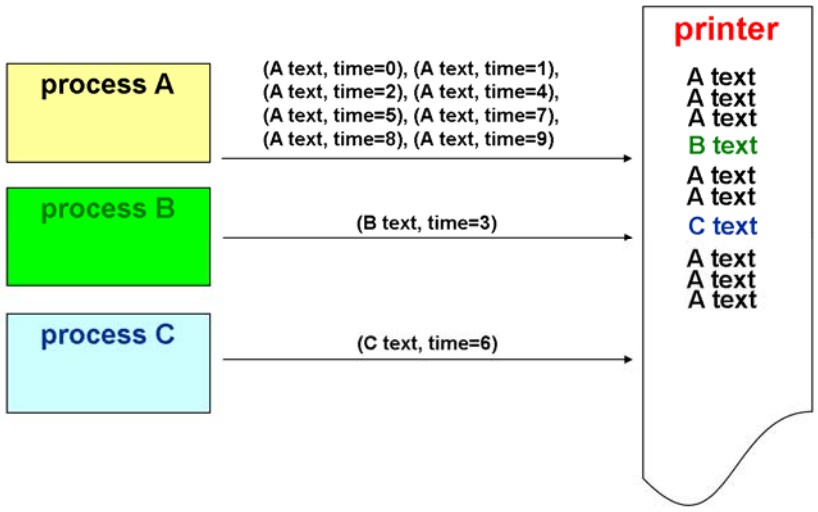


Caching in Kernel Space

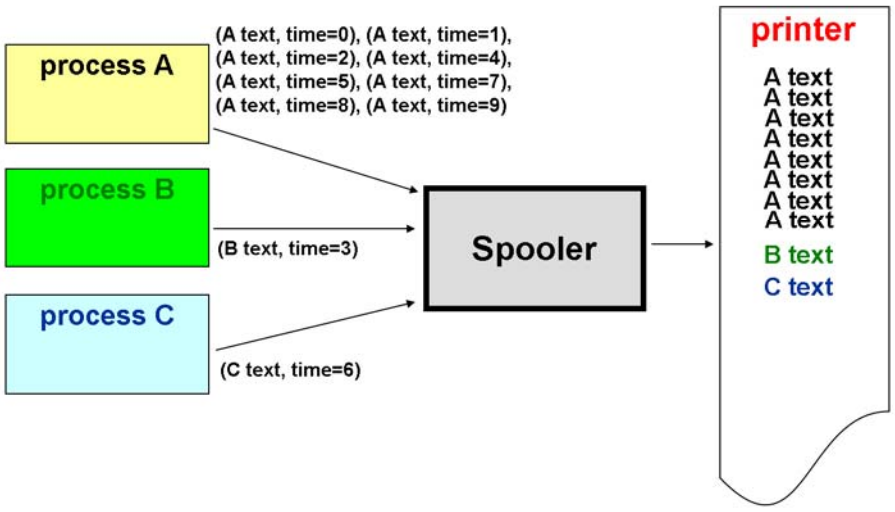


Kernel buffers are used as cache for the disk device.
Consequence: the EAT can be substantially reduced.

Output without Spooling



Output with Spooling



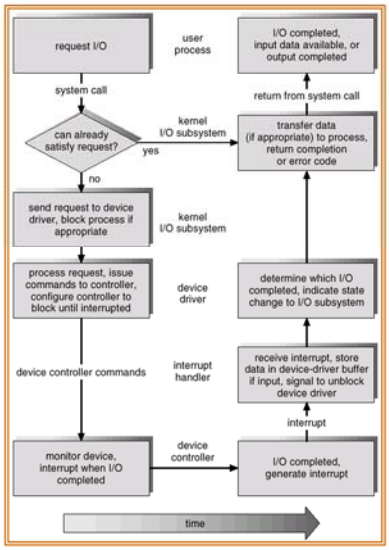
Error Handling

- 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 Requests to Hardware Operations

- Consider reading a file from disk for a process:
  - Determine device holding file.
  - Translate name to device representation.
  - Physically read data from disk into buffer.
  - Make data available to requesting process.
  - Return control to process.

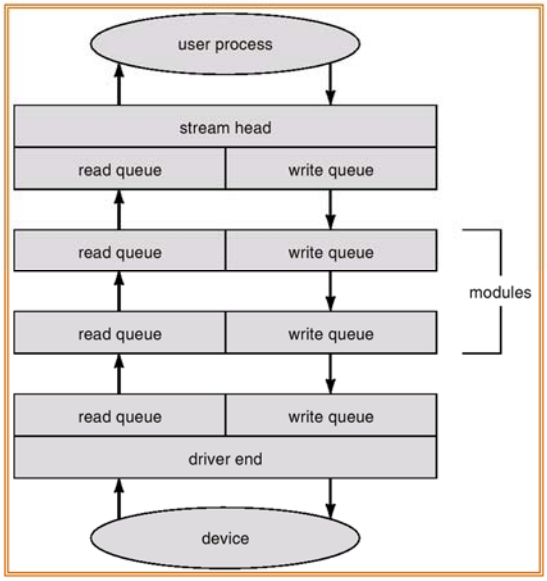
Life Cycle of An I/O Request



STREAMS

- **STREAM** – a full-duplex communication channel between a user-level process and a device.
- A STREAM consists of:
  - STREAM head interfaces with the user process
  - driver end interfaces with the device
  - zero or more STREAM modules between them.
- Each module contains a **read queue** and a **write queue.**
- Message passing is used to communicate between queues.

The STREAMS Structure

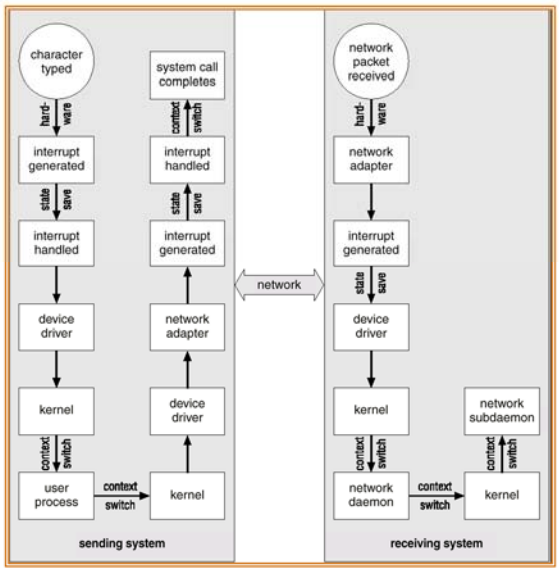


Performance

I/O a major factor in system performance:

- Demands CPU to execute device driver, kernel I/O code.
- Context switches due to interrupts.
- Data copying.
- Network traffic especially stressful.

Intercomputer Communications



Improving Performance

- Reduce number of context switches.
- Reduce data copying.
- Reduce interrupts by using large transfers, smart controllers, polling.
- Use DMA.
- Balance CPU, memory, bus, and I/O performance for highest throughput.

Device-Functionality Progression