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

<table>
<thead>
<tr>
<th>aspect</th>
<th>variation</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-transfer mode</td>
<td>character</td>
<td>terminal</td>
</tr>
<tr>
<td></td>
<td>block</td>
<td>disk</td>
</tr>
<tr>
<td>access method</td>
<td>sequential</td>
<td>modern</td>
</tr>
<tr>
<td></td>
<td>random</td>
<td>CD-ROM</td>
</tr>
<tr>
<td>transfer schedule</td>
<td>synchronous</td>
<td>tape</td>
</tr>
<tr>
<td></td>
<td>asynchronous</td>
<td>keyboard</td>
</tr>
<tr>
<td>sharing</td>
<td>dedicated</td>
<td>tape</td>
</tr>
<tr>
<td></td>
<td>sharable</td>
<td>keyboard</td>
</tr>
<tr>
<td>device speed</td>
<td>latency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seek time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>transfer rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>delay between operations</td>
<td></td>
</tr>
<tr>
<td>I/O direction</td>
<td>read only</td>
<td>CD-ROM</td>
</tr>
<tr>
<td></td>
<td>write only</td>
<td>graphics controller</td>
</tr>
<tr>
<td></td>
<td>read/modify</td>
<td>disk</td>
</tr>
</tbody>
</table>
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/9x/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”.
Buffering in Kernel Space

application

DMA Controller

$y, n$

OS Kernel

buffer (size $n$)

disk

address $x$

buffer (size $n$)

copy semantics respected

04/19/2010

CSCI 315 Operating Systems Design
Caching in Kernel Space

Application

I/O request goes to buffer cache before a disk access is made.

OS Kernel

Disk

Kernel buffers are used as cache for the disk device. Consequence: the EAT can be substantially reduced.
Output without Spooling

process A
(A text, time=0), (A text, time=1),
(A text, time=2), (A text, time=4),
(A text, time=5), (A text, time=7),
(A text, time=8), (A text, time=9)

process B
(B text, time=3)

process C
(C text, time=6)

printer
A text
A text
A text
B text
A text
A text
C text
A text
A text
A text

04/19/2010
CSCI 315 Operating Systems Design
Output with Spooling

process A
(A text, time=0), (A text, time=1),
(A text, time=2), (A text, time=4),
(A text, time=5), (A text, time=7),
(A text, time=8), (A text, time=9)

process B
(B text, time=3)

process C
(C text, time=6)

Spooler

printer
A text
A text
A text
A text
A text
A text
B text
C text
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

- Increased time (generations)
- Increased efficiency
- Increased development cost
- Increased abstraction

new algorithm

- Application code
- Kernel code
- Device-driver code
- Device-controller code (hardware)
- Device code (hardware)

Increased flexibility