

# I/O Systems



## Practice Exercises

- 12.1** State three advantages of placing functionality in a device controller, rather than in the kernel. State three disadvantages.

**Answer:**

Three advantages:

- a. Bugs are less likely to cause an operating-system crash.
- b. Performance can be improved by utilizing dedicated hardware and hard-coded algorithms.
- c. The kernel is simplified by moving algorithms out of it.

Three disadvantages:

- a. Bugs are harder to fix—a new firmware version or new hardware is needed.
- b. Improving algorithms requires a hardware update rather than just a kernel or device-driver update.
- c. Embedded algorithms could conflict with an application's use of the device, causing decreased performance.

- 12.2** The example of handshaking in Section 12.2 used two bits: a busy bit and a command-ready bit. Is it possible to implement this handshaking with only one bit? If it is, describe the protocol. If it is not, explain why one bit is insufficient.

**Answer:**

It is possible, using the following algorithm. Let's assume we simply use the busy bit (or the command-ready bit; this answer is the same regardless). When the bit is off, the controller is idle. The host writes to data-out and sets the bit to signal that an operation is ready (the equivalent of setting the command-ready bit). When the controller is finished, it clears the busy bit. The host then initiates the next operation.

This solution requires that both the host and the controller have read and write access to the same bit, which can complicate circuitry and increase the cost of the controller.

- 12.3** Why might a system use interrupt-driven I/O to manage a single serial port and polling I/O to manage a front-end processor, such as a terminal concentrator?

**Answer:**

Polling can be more efficient than interrupt-driven I/O. This is the case when the I/O is frequent and of short duration. Even though a single serial port will perform I/O relatively infrequently and should thus use interrupts, a collection of serial ports such as those in a terminal concentrator can produce a lot of short I/O operations, and interrupting for each one could create a heavy load on the system. A well-timed polling loop could alleviate that load without wasting many resources through looping with no I/O needed.

- 12.4** Polling for an I/O completion can waste a large number of CPU cycles if the processor iterates a busy-waiting loop many times before the I/O completes. But if the I/O device is ready for service, polling can be much more efficient than catching and dispatching an interrupt. Describe a hybrid strategy that combines polling, sleeping, and interrupts for I/O device service. For each of these three strategies (pure polling, pure interrupts, hybrid), describe a computing environment in which that strategy is more efficient than either of the others.

**Answer:** A hybrid approach could switch between polling and interrupts depending on the length of the I/O operation wait. For example, we could poll and loop  $N$  times, and if the device is still busy at  $N+1$ , we could set an interrupt and sleep. This approach would avoid long busy-waiting cycles and would be best for very long or very short busy times. It would be inefficient if the I/O completes at  $N+T$  (where  $T$  is a small number of cycles) due to the overhead of polling plus setting up and catching interrupts.

Pure polling is best with very short wait times. Interrupts are best with known long wait times.

- 12.5** How does DMA increase system concurrency? How does it complicate hardware design?

**Answer:**

DMA increases system concurrency by allowing the CPU to perform tasks while the DMA system transfers data via the system and memory buses. Hardware design is complicated because the DMA controller must be integrated into the system, and the system must allow the DMA controller to be a bus master. Cycle stealing may also be necessary to allow the CPU and the DMA controller to share use of the memory bus.

- 12.6** Why is it important to scale up system-bus and device speeds as CPU speed increases?

**Answer:**

Consider a system that performs 50% I/O and 50% computes. Doubling the CPU performance on this system would increase total system performance by only 50%. Doubling both system aspects would increase performance by 100%. Generally, it is important to remove the current system bottleneck, and to increase overall system performance, rather than blindly increasing the performance of individual system components.

- 12.7 Distinguish between a driver end and a stream module in a STREAMS operation.

**Answer:**

The driver end controls a physical device that could be involved in a STREAMS operation. The stream module modifies the flow of data between the stream head (the user interface) and the driver.

