tion. Later, when the device finishes its current operation and generates an interrupt, the lower half extracts the next request from the queue, starts the device, and returns from the interrupt. Figure 17.3 illustrates the conceptual organization.

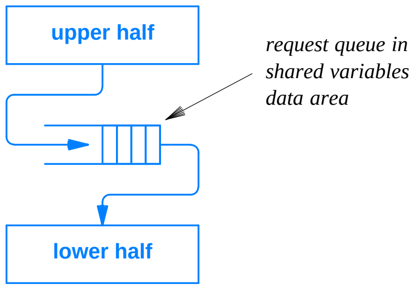

**Figure 17.3** The conceptual organization of a device driver that uses a queue of requests. On output, the upper half deposits items in the request queue without waiting for the device, and the lower half controls the device.

A device driver that uses an output queue is elegant — the queue of requests provides coordination between the upper and lower halves of the driver. Figure 17.4 lists the steps that each half of a device driver takes for output.

**Initialization (computer system starts)**

1. Initialize output queue to empty

**Upper half (application performs write)**

1. Deposit data item in queue
2. Use the CSR to request an interrupt
3. Return to application

**Lower half (interrupt occurs)**

1. If the queue is empty, stop the device from interrupting
2. If the queue is nonempty, extract an item and start output
3. Return from interrupt

**Figure 17.4** The steps that the upper and lower halves of a device driver take for an output operation when queueing is used. The upper half forces an interrupt, but does not start output on the device.