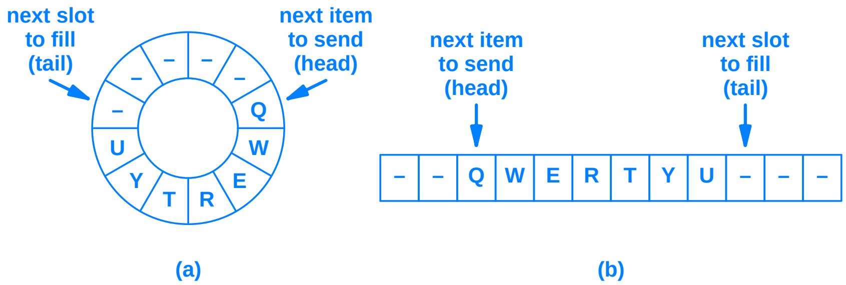

Figure 15.2 (a) A circular output buffer acting as a queue, and (b) the implementation with an array of bytes.

Output functions deposit characters to be sent in the output buffer and return to their caller. When it places characters in the output buffer, an upper-half function must also start output interrupts on the device. Whenever the device generates an output interrupt, the lower half extracts up to sixteen characters from the output buffer, and deposits the characters in the device's output FIFO.† Once all characters in the output FIFO have been transmitted, the device will interrupt again. Thus, output continues until the output buffer becomes empty at which time the driver stops output and the device becomes idle.

Input works the other way around. Whenever it receives characters, the device interrupts and the interrupt dispatcher calls a lower-half function (i.e., *ttyhandler*). The interrupt handler extracts the characters from the device's input FIFO and deposits them in the circular input buffer. When a process calls an upper-half function to read input, the upper-half function extracts characters from the input buffer.

Conceptually, the two halves of a driver only communicate through shared buffers. Upper-half functions place outgoing data in a buffer and extract incoming data from a buffer. The lower half extracts outgoing data from the buffer and sends it to the device, and places incoming data in the buffer. To summarize:

> *Upper-half functions transfer data between processes and buffers; the lower half transfers data between buffers and the device hardware.*

15.6 Synchronization Of Upper Half And Lower Half

In practice, the two halves of the driver usually need to do more than manipulate a shared data structure. For example, an upper-half function may need to start an output transfer if a device is idle. More important, the two halves need to coordinate operations on the request queue and the buffers. For example, if all slots in the output buffer are full when a process tries to write data, the process must be blocked. Later, when characters have been sent to the device and buffer space becomes available, the blocked

†To improve efficiency, most UART hardware has a small on-board character buffer that can hold up to 16 outgoing characters at a time.