calling function *erase1* to erase the character from the display. When the line kill character, *tyikillc*, arrives, *ttyhandle_in* eliminates the current line by setting *tyicursor* to zero and moving the tail pointer back to the beginning of the line. Finally, when a *NEWLINE* or *RETURN* character arrives, *ttyhandle_in* calls *signaln* to make the entire input line available. It resets *tyicursor* to zero for the next line. Note that the test for buffer full always leaves one extra space in the buffer for the end-of-line character (i.e., a *NEWLINE*).

15.19 Tty Control Block Initialization (ttyinit)

Function *ttyinit*, shown below in file *ttyinit.c*, is called once for each tty device. The parameter, *devptr*, is a pointer to the device switch table entry for a tty device. *Ttyinit* extracts the minor device number from the device switch table, uses the number as an index into the *ttytab* array, and sets *typtr* to the tty control block for the device. *Ttyinit* then initializes each field in the control block, setting the device to cooked mode, creating the input and output semaphores, and assigning head and tail pointers to indicate that buffers are empty. After driver parameters and buffers have been initialized, *ttyinit* sets variable *uptr* to the CSR address of the UART hardware. It then sets the baud rate, sets the bits per character, turns off hardware parity checking, and sets the number of RS-232 stop bits to 1. Finally, *ttyinit* disables transmitter interrupts temporarily.

Once the basic hardware initialization has been completed, *ttyinit* calls *set_evec* to set the interrupt vector to the interrupt function given in the device switch table. After the interrupt vector has been set, *ttyinit* finishes the last steps of hardware initialization: it resets the hardware to permit both transmit and receive interrupts, and calls *ttykickout* to start output.

Ttyinit assumes a tty will be associated with a keyboard and display that a human will use. Consequently, *ttyinit* initializes a tty to cooked mode, and sets parameters assuming a video device that can backspace over characters on the display and erase them. In particular, the parameter *tyieback* causes *ttyhandle_in* to echo three characters, backspace-space-backspace, when it receives the erase character, *tyierasec*. On a display screen, sending the three-character sequence gives the effect of erasing characters as the user backs over them. If you look again at *ttyhandle_in*,† you will see that it carefully backs up the correct number of spaces, even if the user erases a control character that has been displayed as two printable characters.

†The code for *ttyhandle_in* can be found on page 308.