# The Lockronomicon

Your guide to the ancient and twisted locking policies of the tty layer and the warped logic behind them. Beware all ye who read on.

FIXME: still need to work out the full set of BKL assumptions and document them so they can eventually be killed off.

## Line Discipline

Line disciplines are registered with tty_register_ldisc() passing the discipline number and the ldisc structure. At the point of registration the discipline must be ready to use and it is possible it will get used before the call returns success. If the call returns an error then it won't get called. Do not re-use ldisc numbers as they are part of the userspace ABI and writing over an existing ldisc will cause demons to eat your computer. After the return the ldisc data has been copied so you may free your own copy of the structure. You must not re-register over the top of the line discipline even with the same data or your computer again will be eaten by demons.

In order to remove a line discipline call tty_unregister_ldisc().
In ancient times this always worked. In modern times the function will return -EBUSY if the ldisc is currently in use. Since the ldisc referencing code manages the module counts this should not usually be a concern.

Heed this warning: the reference count field of the registered copies of the tty_ldisc structure in the ldisc table counts the number of lines using this discipline. The reference count of the tty_ldisc structure within a tty counts the number of active users of the ldisc at this instant. In effect it counts the number of threads of execution within an ldisc method (plus those about to enter and exit although this detail matters not).

## Line Discipline Methods

TTY side interfaces:

| | | |
|---|---|---|
| open() | - | Called when the line discipline is attached to the terminal. No other call into the line discipline for this tty will occur until it completes successfully. Returning an error will prevent the ldisc from being attached. Can sleep. |
| close() | - | This is called on a terminal when the line discipline is being unplugged. At the point of execution no further users will enter the ldisc code for this tty. Can sleep. |
| hangup() | - | Called when the tty line is hung up. The line discipline should cease I/O to the tty. No further calls into the ldisc code will occur. The return value is ignored. Can sleep. |

| | | |
|---|---|---|
| write() | - | A process is writing data through the line discipline.  Multiple write calls are serialized by the tty layer for the ldisc.  May sleep. |
| flush_buffer() | - | (optional) May be called at any point between open and close, and instructs the line discipline to empty its input buffer. |
| chars_in_buffer() | - | (optional) Report the number of bytes in the input buffer. |
| set_termios() | - | (optional) Called on termios structure changes. The caller passes the old termios data and the current data is in the tty. Called under the termios semaphore so allowed to sleep. Serialized against itself only. |
| read() | - | Move data from the line discipline to the user. Multiple read calls may occur in parallel and the ldisc must deal with serialization issues. May sleep. |
| poll() | - | Check the status for the poll/select calls. Multiple poll calls may occur in parallel. May sleep. |
| ioctl() | - | Called when an ioctl is handed to the tty layer that might be for the ldisc. Multiple ioctl calls may occur in parallel. May sleep. |
| compat_ioctl() | - | Called when a 32 bit ioctl is handed to the tty layer that might be for the ldisc. Multiple ioctl calls may occur in parallel. May sleep. |

Driver Side Interfaces:

| | | |
|---|---|---|
| receive_buf() | - | Hand buffers of bytes from the driver to the ldisc for processing. Semantics currently rather mysterious 8( |
| write_wakeup() | - | May be called at any point between open and close. The TTY_DO_WRITE_WAKEUP flag indicates if a call is needed but always races versus calls. Thus the ldisc must be careful about setting order and to handle unexpected calls. Must not sleep. |
| | | The driver is forbidden from calling this directly from the ->write call from the ldisc as the ldisc is permitted to call the driver write method from this function. In such a situation defer it. |
| dcd_change() | - | Report to the tty line the current DCD pin status changes and the relative timestamp. The timestamp can be NULL. |

## Driver Access

Line discipline methods can call the following methods of the underlying hardware driver through the function pointers within the tty->driver structure:

| | |
|---|---|
| write() | Write a block of characters to the tty device. Returns the number of characters accepted. The character buffer passed to this method is already in kernel space. |
| put_char() | Queues a character for writing to the tty device. If there is no room in the queue, the character is ignored. |
| flush_chars() | (Optional) If defined, must be called after queueing characters with put_char() in order to start transmission. |
| write_room() | Returns the numbers of characters the tty driver will accept for queueing to be written. |
| ioctl() | Invoke device specific ioctl. Expects data pointers to refer to userspace. Returns ENOIOCTLCMD for unrecognized ioctl numbers. |
| set_termios() | Notify the tty driver that the device's termios settings have changed. New settings are in tty->termios. Previous settings should be passed in the "old" argument.<br><br>The API is defined such that the driver should return the actual modes selected. This means that the driver function is responsible for modifying any bits in the request it cannot fulfill to indicate the actual modes being used. A device with no hardware capability for change (eg a USB dongle or virtual port) can provide NULL for this method. |
| throttle() | Notify the tty driver that input buffers for the line discipline are close to full, and it should somehow signal that no more characters should be sent to the tty. |
| unthrottle() | Notify the tty driver that characters can now be sent to the tty without fear of overrunning the input buffers of the line disciplines. |
| stop() | Ask the tty driver to stop outputting characters to the tty device. |
| start() | Ask the tty driver to resume sending characters to the tty device. |
| hangup() | Ask the tty driver to hang up the tty device. |

| | |
|---|---|
| break_ctl() | (Optional) Ask the tty driver to turn on or off BREAK status on the RS-232 port.  If state is -1, then the BREAK status should be turned on; if state is 0, then BREAK should be turned off. If this routine is not implemented, use ioctls TIOCSBRK / TIOCCBRK instead. |
| wait_until_sent() | Waits until the device has written out all of the characters in its transmitter FIFO. |
| send_xchar() | Send a high-priority XON/XOFF character to the device. |

## Flags

Line discipline methods have access to tty->flags field containing the following interesting flags:

| | |
|---|---|
| TTY_THROTTLED | Driver input is throttled. The ldisc should call tty->driver->unthrottle() in order to resume reception when it is ready to process more data. |
| TTY_DO_WRITE_WAKEUP | If set, causes the driver to call the ldisc's write_wakeup() method in order to resume transmission when it can accept more data to transmit. |
| TTY_IO_ERROR | If set, causes all subsequent userspace read/write calls on the tty to fail, returning -EIO. |
| TTY_OTHER_CLOSED | Device is a pty and the other side has closed. |
| TTY_NO_WRITE_SPLIT | Prevent driver from splitting up writes into smaller chunks. |

## Locking

Callers to the line discipline functions from the tty layer are required to take line discipline locks. The same is true of calls from the driver side but not yet enforced.

Three calls are now provided

```
        ldisc = tty_ldisc_ref(tty);
```

takes a handle to the line discipline in the tty and returns it. If no ldisc is currently attached or the ldisc is being closed and re-opened at this point then NULL is returned. While this handle is held the ldisc will not change or go away.

```
        tty_ldisc_deref(ldisc)
```

Returns the ldisc reference and allows the ldisc to be closed. Returning the reference takes away your right to call the ldisc functions until you take a new reference.

```
        ldisc = tty_ldisc_ref_wait(tty);
```

Performs the same function as tty_ldisc_ref except that it will wait for an ldisc change to complete and then return a reference to the new ldisc.

While these functions are slightly slower than the old code they should have minimal impact as most receive logic uses the flip buffers and they only need to take a reference when they push bits up through the driver.

A caution: The ldisc->open(), ldisc->close() and driver->set_ldisc functions are called with the ldisc unavailable. Thus tty_ldisc_ref will fail in this situation if used within these functions. Ldisc and driver code calling its own functions must be careful in this case.

## Driver Interface

| | | |
|---|---|---|
| open() | - | Called when a device is opened. May sleep |
| close() | - | Called when a device is closed. At the point of return from this call the driver must make no further ldisc calls of any kind. May sleep |
| write() | - | Called to write bytes to the device. May not sleep. May occur in parallel in special cases. Because this includes panic paths drivers generally shouldn't try and do clever locking here. |
| put_char() | - | Stuff a single character onto the queue. The driver is guaranteed following up calls to flush_chars. |
| flush_chars() | - | Ask the kernel to write put_char queue |
| write_room() | - | Return the number of characters tht can be stuffed into the port buffers without overflow (or less). The ldisc is responsible for being intelligent about multi-threading of write_room/write calls |
| ioctl() | - | Called when an ioctl may be for the driver |
| set_termios() | - | Called on termios change, serialized against itself by a semaphore. May sleep. |
| set_ldisc() | - | Notifier for discipline change. At the point this is done the discipline is not yet usable. Can now sleep (I think) |
| throttle() | - | Called by the ldisc to ask the driver to do flow control.  Serialization including with unthrottle is the job of the ldisc layer. |
| unthrottle() | - | Called by the ldisc to ask the driver to stop flow control. |

```
stop()           -       Ldisc notifier to the driver to stop output. As with
                         throttle the serializations with start() are down
                         to the ldisc layer.

start()          -       Ldisc notifier to the driver to start output.

hangup()         -       Ask the tty driver to cause a hangup initiated
                         from the host side. [Can sleep ??]

break_ctl()      -       Send RS232 break. Can sleep. Can get called in
                         parallel, driver must serialize (for now), and
                         with write calls.

wait_until_sent() -      Wait for characters to exit the hardware queue
                         of the driver. Can sleep

send_xchar()      -      Send XON/XOFF and if possible jump the queue with
                         it in order to get fast flow control responses.
                         Cannot sleep ??
```