TTY

Teletypewriter (TTY) layer takes care of all those serial devices. Including the virtual ones like pseudoterminal (PTY).

TTY structures

There are several major TTY structures. Every TTY device in a system has a corresponding struct tty_port. These devices are maintained by a TTY driver which is struct tty_driver. This structure describes the driver but also contains a reference to operations which could be performed on the TTYs. It is struct tty_operations. Then, upon open, a struct tty_struct is allocated and lives until the final close. During this time, several callbacks from struct tty_operations are invoked by the TTY layer.

Every character received by the kernel (both from devices and users) is passed through a preselected :doc:`tty_ldisc` (in short ldisc; in C, struct tty_ldisc_ops). Its task is to transform characters as defined by a particular ldisc or by user too. The default one is n_tty, implementing echoes, signal handling, jobs control, special characters processing, and more. The transformed characters are passed further to user/device, depending on the source.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\tty\(linux-master) (Documentation) (tty)index.rst`, line 21); *backlink***
>
> Unknown interpreted text role "doc".

In-detail description of the named TTY structures is in separate documents:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\tty\(linux-master) (Documentation) (tty)index.rst`, line 31)**
>
> Unknown directive type "toctree".
>
> ```
> .. toctree::
>    :maxdepth: 2
>
>    tty_driver
>    tty_port
>    tty_struct
>    tty_ldisc
>    tty_buffer
>    n_tty
>    tty_internals
> ```

Writing TTY Driver

Before one starts writing a TTY driver, they must consider :doc:`Serial <../driver-api/serial/driver>` and :doc:`USB Serial <../usb/usb-serial>` layers first. Drivers for serial devices can often use one of these specific layers to implement a serial driver. Only special devices should be handled directly by the TTY Layer. If you are about to write such a driver, read on.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\tty\(linux-master) (Documentation) (tty)index.rst`, line 45); *backlink***
>
> Unknown interpreted text role "doc".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\tty\(linux-master) (Documentation) (tty)index.rst`, line 45); *backlink***
>
> Unknown interpreted text role "doc".

A *typical* sequence a TTY driver performs is as follows:

1. Allocate and register a TTY driver (module init)
2. Create and register TTY devices as they are probed (probe function)
3. Handle TTY operations and events like interrupts (TTY core invokes the former, the device the latter)
4. Remove devices as they are going away (remove function)
5. Unregister and free the TTY driver (module exit)

Steps regarding driver, i.e. 1., 3., and 5. are described in detail in :doc:`tty_driver`. For the other two (devices handling), look into :doc:`tty_port`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\tty\(linux-master) (Documentation) (tty)index.rst`, line 61); *backlink*

Unknown interpreted text role "doc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\tty\(linux-master) (Documentation) (tty)index.rst`, line 61); *backlink*

Unknown interpreted text role "doc".