"Terminals" are one of the earliest established standards for human-computer interaction.

Some of the nitty-gritty details about terminals are only of historical significance, but modern POSIX text shell ("commandline") interfaces are still built on terminal principles.

In COS222 we will not delve into the details of the Minix tty driver itself.

But it is important to at least understand what a virtual tty is, how the concept fits into the operating system as a whole, and why the Minix tty driver handles both keyboard and screen.
Terminals

terminal (noun): an end point of some system
(from Latin terminus, "a bound, boundary, limit, end")

In telecommunications, terminal means: the apparatus to send and/or receive signals on a line, such as a telephone or network device.
   In other words, the device at one end of the wire/communications line.

In computing: A device for entering data into a computer or a communications system and/or displaying data received, especially a device equipped with a keyboard and some sort of textual display.
   In other words the device at the user's end of the system.

The terminal driver (src/drivers/tty) handles keyboard and screen in Minix.
This might seem odd as the devices are actually separate.

Originally, keyboards and screens were not separate devices.
Instead, the user interacted with the computer using a terminal.

Although the physical terminal technology of 1940-1980 is rarely seen today, terminals still exist as an important concept on Unix-like operating systems.
A terminal consists of a text display and text input device.
Teleprinter terminals

The word "tty" is often used to refer to a terminal. It was an abbreviation for Teletype Corporation, one of the leading companies that manufactured teleprinter terminals from the 1930's to 1980's. The brand became so popular that people started referring to any teleprinter as a "teletype" or "TTY".

TTYs were used before modern programmable microcomputers to send information over long distances. The earliest computers did not work with keyboards or screens (other devices such as punch cards were used).

However, computer systems developers realised that textual input and output devices could be very useful, and soon developed interfaces for teleprinter terminals.

Later, electronic TTYs were developed. These used Cathode Ray Tube displays with a keyboard.
TTY terminals later became the default way for users to interact with mainframe computer systems.

Usually, many users would be connected to a mainframe simultaneously, each via their own TTY. The terminal would often be some distance away from the actual computer.

With the advent of personal computers, it became clear that separating the keyboard and screen devices is a better idea.

However, the concept of a TTY was by then an integral part of most operating systems.

TTY-like IO still remains the most basic way in which a user can interface with a personal computer, and almost all modern display and keyboard controllers can be operated in this way by the OS.
Terminals in modern software

The Display device controller on a PC supports several display modes, for example:

- 80x25 text mode
- 640x480 pixels graphical mode with 8-bit color
- 1360x720 pixels 24bit color graphical mode

Most of the time, high-resolution color graphical mode is used. So why keep things like text mode in modern computers?

- The drivers for graphical modes are vastly more complex than the drivers for TTY-like text mode.
- Almost all display controllers will support some text mode, no matter how primitive. Text displays provide a reliable fallback.
- It is generally a good idea to initialize drivers "as late as possible". If the OS fails for some reason during startup, it must be possible for it to communicate this to the user without needing a complex graphical display.
- It is much simpler (and faster) to communicate with a remote computer's virtual TTY interface than over a graphical remote-desktop access tool such as VNC.
Virtual TTYs

Any POSIX-based system has multiple virtual terminals. Each one is viewed as a separate TTY by the OS. These can always be accessed, even if the graphical interface stops working for some reason.

The user can switch between consoles using Ctrl+Alt+F1 to Ctrl+Alt+F8.

When you open an XTerminal window, it creates and connects to another "fake" tty (called a pseudo-tty).

The same is done for remote logins over Telnet or SSH.

SSH (Secure Shell) is a popular tool for logging into computers from another over a LAN or the Internet. A graphical display can work over SSH, but in its most basic form SSH provides a TTY-style text interface.
Graphical window systems don't use the screen and keyboard as a TTY. Therefore the drivers for these devices must support multiple modes. User software (such as the X11 Window System commonly used on Linux) can request the OS to switch the display driver from text mode to graphical mode. The display driver can work in both modes at once, since it handles the bitmap display as well as virtual terminals. The diagram represents the overall idea. But keep in mind: reality is often more complex. Modern display controllers ("graphics cards") usually have their own separate drivers in the OS, although almost all can be switched to text mode.
Terminals can be connected to a system in various ways. The screen+keyboard "terminal" is a memory-mapped IO device. When the display driver is in text mode, it will expose "video RAM" to the main RAM bus. Programs can write to these areas in memory to display output.

In the past, the RS232 serial port was commonly used to connect terminals (and other devices). The "device controller" for such a terminal is a Universal Asynchronous Receiver/Transmitter chip.

RS232 is not common anymore, but UARTs are used a lot in computer engineering when working with serial communications networks.
TTY modes

Various different text modes were developed for teletype terminals. Some of these modes allowed the driver to set additional properties for certain characters on the screen, eg. colors, bold, underlined, flashing text, etc. Additionally, certain modes made it possible for user-space software to move the cursor around on the screen and update parts of it. This allowed the implementation of more advanced text editors and user interfaces.

Two modes were eventually standardized: "raw mode" and "cooked mode"

- In "cooked mode" (aka canonical mode) the user program does not have to deal with the complexities of the terminal, and the driver only allows it to read whole lines of input at a time (think of stdin/cin in C/C++) or to write characters.
- In "raw mode" the tty driver gives more control to the user program, making it possible to blank the screen, place text at specific coordinates, etc.

Handling raw mode can get complex, so libraries were developed to make it easier, such as **ncurses** ("new cursor optimization library"). These are still actively used in commandline OS configuration tools, Nvidia's Linux driver installation tool, and games such as Dungeon Crawl and Brogue.
To enable colors, bold, etc. a terminal would typically have more than one byte in memory per character: eg. one byte for storing attributes and another for the character itself.

The driver needs some way to indicate to the TTY that a "special" operation should be done, such as moving the cursor, or using a color.

This is usually done by writing an "escape character" (ASCII 0xB1) followed by certain operation codes.

The kinds of operation codes recognized depend on whether the terminal is in raw or cooked mode. Virtual TTYs on Unix-like operating systems usually also support these display modes. Color is often used to make text shells more user-friendly.