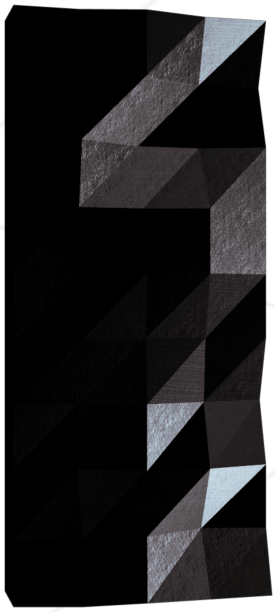

Let's talk about termcaps

ft_select

Summary: Small introduction to termcaps.

Contents

Chapter I

Termcaps

`ft_select` overuses the `termcaps` library. But what are `termcaps` exactly? `Termcap` refers to your terminal's "habilities" (moving the cursor, deleting a line, etc.) You could see them as the magic tricks that your terminal can do. Since the invention of the terminal (this happened a long time ago in a galaxy far, far away), one could find a fuckton of those termcaps, each with their own, more or less standard features, and implementation methods. In other words, it was a mess and every program looking to be portable was considered a joke.

It was only then that a group of bearded wannabe super heros decided to clear up that mess. All the terminals were recorded and catalogued with a list of their features in order to get a reliable database with all the available features. They didn't stop there. The bearded guys decided to create a library capable of documenting a terminal's available features and use them in a unified manner. So awesome! Just amazing! And this is the story of the `termcaps` library that you'll be using for the next couple of projects.

`The text above is pure fiction and is in no way related to reality. But it allows for a useful chronology and introduction to termcaps. You can ask google for the real story. However, if any of you has any drawing talent, a stip illustrating this fiction would be greatly appreciated and would be added to this document.`

Once you have found your terminal in the database using the function `tgetent(3)`, its features are represented by booleen flags, integers and strings. All you need to do then is to use the feature's associated function to request the database: `tgetflag(3)`, `tgetnum(3)` and `tgetstr(3)`. All those features have a unique 2-character code representing them. For instance, if you wish to know how many arms you terminal has, you could write:

```
int     nbarms;

nbarms = tgetnum("ar");
```

Don't even try to compile this example because up to this day, no terminal with arms has been recorded, and the `tgetnum` most likely will return `-1`, unless the `"ar"` feature really exists for something otherwise more serious.

Some features are activated when writing a series of specific characters in the terminal. For you to know what string is required, you need to interrogate the database using `tgetstr(3)` and write the result in the terminal using `tputs(3)`. If your terminal suddenly transforms itself into an abstract painting, then you most likely did something wrong.

You have to understand that some of your terminal's features are used by writing a string in your terminal and some others by comparing the string received from the keyboard to the feature's string. Take the arrow keys as an example. Some keys or key combinations aren't represented by a single character. This is the case for arrow keys. It's really important for you to know that. Therefore, if you want to move the cursor one column to the right, you can't write the "right arrow" feature in your terminal. You have to get the characters representing "right arrow" sent by the keyboard, then compare them to your terminal's "right arrow" feature string. If they're the same, then you can write the "move the cursor a column to the right" feature into your terminal. Does it make sense?

Let's be honest, termcaps aren't intuitive. That's why reading their official documentation should prove helpful to you. Take the time to test and share your mistakes and challenges.

The question of you're terminal basic configuration also arises. Up until now, when you pressed `a` on your keyboard, you weren't surprised to see `'a'` displayed on your terminal. You were not surprised to see your terminal send the line to your `shell` only after you pressed `return`. Yet, you're not really looking forward to having you program behave in such a way. You will need to fix this by putting your terminal into "raw" or "non-canonical" mode. Google is your best friend. Note that `tcgetattr(3)` and `tcsetattr(3)` functions will be helpful. Study the behavior of programs like `herrie` or `mcabber` to get a better idea of what i mean by that.

Anyways, `termcaps` adventure start with the collective and deep reading of those `man`:

- termios
- termcap

These man are long and difficult to read! Be smart! Don't take yourself for a scholar. Search for relevant information from which you can extrapolate! Don't read them alone! Create a group with your peers, read the man aloud, discuss them and take notes if necessary. Reading these man alone will lead you to feel small and meaningless in the immensity of the universe.