L2/25-263

Text Terminal Working Group report

Author: Fraser Gordon (chair)

The Text Terminal Working Group (TTWG) exists to improve the support of using complex scripts (e.g. those requiring shaping, or “wide” characters which don’t fit legibly into a character cell appropriate for Latin text) in the fixed-width rendering context of text terminals. Besides native language support, programs are making increasing use of emoji in terminal-based applications, with inconsistent results from different terminal applications.

TTWG has consensus within the group as to the general approach to solving these problems. The core part of this is supporting two classes of terminal software: those with simple rendering needs, and those with complex rendering needs. Simple applications treat the terminal as a line-based I/O system, without making deliberate use of cursor positioning or other escape sequences, and expect the terminal to handle layout and rendering. The prototypical example would be `cat`-ing a text file to the terminal and having it display correctly.

Complex applications are those that explicitly treat the terminal as a 2-dimensional character cell grid and expect to position text and/or graphical elements (such as box-drawing characters) at particular locations within the grid. Examples are text editors (e.g. `vim`, `emacs`), terminal multiplexers (e.g. `tmux`, `screen`) and applications that present a TUI (e.g. by using `ncurses` to present a user interface). These applications need to work with the terminal and share responsibility for layout.

An example of a common pattern in complex applications is having a fixed-width area into which text will be drawn. This could be a field for text entry, the caption on a button, or a label beside a checkbox. For this to work without visual corruption, the application and the terminal have to have the same understanding of how many cells a given piece of text will occupy.

To achieve this understanding, TTWG will define measurement rules that define the number of character cells that will be occupied by a string. This measurement is independent of font metrics; our design is that fonts that wish to be compatible will adapt their metrics to match these rules.

We have identified that the nature of the problem is in some ways similar to support for bidirectional text, in particular that context beyond a single character cell is needed for analysis. Modifying a single character may require reflow and/or redraw of the entire string, more similar to variable-width text layout than fixed-width. In contrast to bidi, context for measurement can be limited to that needed for shaping, and not entire paragraphs. Note that shaping is needed before measurement, since it affects grapheme sizes.

Support for bidirectional text in terminals is far from universal. Some frequently-used terminals have very limited support or have had recent regressions in support. As such, there is a desire in the group to not be dependent on terminals supporting UAX 9. This is an area for further discussion, including for a more general ability of terminals and applications to indicate partial support for complex text layout (e.g. support for wide characters but not for text shaping).