1. Turtle. Turtle is a modern emulator of a computer terminal. A computer terminal is a device which communicates with a computer. In the earliest days of automatic computing machines, as they were then known[1], the computer was an enormous machine and terminals had little or no processing ability of their own.

The first devices that can be properly thought of as terminals consisted of a printer and a typewriter keyboard. Rather than marking the paper like a plain typewriter would the terminal informs the computer (host) which key was pressed and the host can instruct the printer to print. They were known as teletypewriters and although they have long since become obsolete the communication line that modern terminals use is still called a TTY, or a pseudo-TTY or PTY if the link is simulated between two computer programs.

Attempts to define the communication between the terminal and the host began long before computers existed. Shortly after the invention of the electric telegraph, in the 1870s, Émile Baudot invented a method of encoding information that used a fixed number of bits for each item of data. This system lent itself better to processing by machines than Morse code and soon replaced it.

Baudot's first code used six bits, far more than necessary for the basic English alphabet, so the next iteration which was popularised and later became known as International Telegraph Alphabet 1 (ITA1) reduced the space to five bits.

Five bits allows only 32 distinct symbols to be encoded so Baudot/ITA1 defined two codes which change the meaning of all subsequent codes until reaching another changing code: `-O.---` changes meaning to figures, `O-.---` changes to letters. This is now known as shifting, which has fortunately been made mostly obsolete by Unicode.

Baudot's code was designed to be typed into a simple keyboard with two keys controlled by the left hand and three by the right, corresponding to the position of each of the five bits. Towards the end of the century typewriters became popular and Donald Murray developed a system which encoded each key's combination as holes punched into a strip of paper tape. With this improvement a new code was developed that optimised the arrangement of bits for the benefit of the machinery (eg. by encoding common letters with fewer holes to punch) rather than for operator memory. This code with some changes eventually became known as ITA2.

Murry's code is also a 5-bit code with two shift levels for figures and letters. It is important to note that some codes (in particular the codes that perform the shift) are the same in both levels.

Interestingly Murray's original printer was criticised for requiring manual operation, so the next machine he produced had a "line" control which advanced the paper (Line Feed) and returned the print head to the margin (Carriage Return). This iteration also introduced the concept of using the code with all bits set to erase the previous code by using that value to (re-)introduce the letters shift level.

After many years of use and growth and FIELDATA (which also used codes of 6 and 7 bits), and despite IBM's attempt to turn it into EBCDIC, national standards bodies eventually ratified the 7-bit standard code for information interchange into ANSI-X3.4 (it was known as ASA then, not ANSI), ECMA-6 and ISO-646, known as ASCII. This code rearranged the letters and symbols again, this time into alphabetical and numeric order.

ASCII expanded the concepts of level shifting and line motions into a set of control codes that ended up in the first two columns, sometimes called sticks, of the code table. ASCII was designed to allow national bodies to define their own variant with custom characters in chosen locations. This thankfully saw very little uptake but a similar technique based more closely on ITA 1 & 2 level shifting was adopted by expanding the code to eight bits and designating codes or code sequences to select from one of several 7-bit character sets.

Like with ASCII the standards bodies worked co-operatively with only a little undue pressure from the Americans and/or British to further develop character encoding techniques and so each organisation has ratified each others' standards, giving them their own identifier.

Because of reasons some of those organisations protect the published standard documents behind strict copyright and fees while others release theirs for free. Sometimes the draft of a published standard, practically identical to the finished product, are made available from the same organisation that protects the final publication.

We will be referring to the free standards from ECMA.

[1] A "computer" was a person who would calculate, or compute, the answer to a mathematical problem.

2. These are the standards, including some pseudo or de-facto standards, which govern a terminal's communication with its host:

ECMA-6, known as ISO-646, ANSI-X3.4 or ASCII. 7-Bit Coded Character Set.

ECMA-35, ISO-2022 or ANSI-X3.41: Character Code Structure and Extension Techniques.

ECMA-43, or ISO-4873: 8-Bit Coded Character Set Structure and Rules.

ECMA-48, ISO-6429 or ANSI-X3.64: Control Functions for Coded Character Sets.

Turtle also supports Unicode, also known as ISO-10646.

Digital Equipment Corporation made a series of standard documents describing the operation of their terminals and terminal emulators which Turtle mostly emulates. These are known as DEC-STD-070.

The development of personal computers led Xterm to become the de facto standard terminal emulator and it includes a document describing the control sequences it understands known as ctlseqs.

The contents of a terminal description file (terminfo) are described by ncurses in the file Caps.

3. Turtle is a unix application with few dependencies however it has been written with an eye towards portability:

OpenGL graphics on X (GLX). Turtle uses libepoxy[1] to find extensions and resolve symbols.

libevent[2] to multiplex I/O.

Portable Snippets[3]. A collection of C preprocessor macros that aid portability, included in the `parsnip` directory.

And of course `CWEB`[4] is required to build the sources.

4. Turtle is built from several modules. The most significant are `panel.o` which draws the visual display and `term.o` which handles communication with the host. These comprise the implementation of the interchange standards discussed above. In addition there is `turtle.o` which provides them with a user interface. Other modules have a support role:

`atlas.o` — texture atlas
`buf.o` — dynamic C buffer
`con.o` — a command-and-control interface
`font.o` — hardly rendering fonts
`glooey.o` — experimental gui framework
`jj.o` — atoms and s-expression evaluator (scheme)
`site.o` — C preprocessor abuse and portability hacks
`tap.o` — test suite
`tri.o` — OpenGL/GLX interface
`ucpdb.o` — database for per-code-point information
`unicode.o` — utf-8 codec

5. The brief history of terminal communication techniques was significantly informed by The Evolution of Character Codes, 1874–1968 by Eric Fischer which also contains a wealth of references to further information.

INTRO