Contents

Encodings

In digital typography there are two seperate (but related) mapping tables. The first is character encoding, which maps a grapheme to a codepoint (ie integer). The second maps a codepoint to a font glyph.

Telegraphy

With the invention of the telegraph in the 1830s came the requirement to encoding text, in this case the latin alphabet and arabic numerals, into electical signals. Early examples of the the the Cooke and Wheatstone system and later the Morse code.

Later telegraph systems used five wires and therefore could sent one of 2^5 different patterns (each pattern is known as a symbol). The original code was invented by Baudot in 1870 and became the International Telegraph Alphabet 1. The limitation of 32 symbols meant it did not contain the numerals.

In 1923 the International Telegraph Alphabet 2 (ITA2) was standarized by the CCITT. This allowed for 62 symbols by using two shift symbols to change the code page, allowing for numerals and punctuation to be sent.

7-bit ASCII

In the 1960s work was done to create a replacement code using 7 bits, allowing for 128 symbols. This allowed for three copies of the latin alphabet (upper case, lower case and control characters), as well as the numerals and a wide range of punctuation.

Around the same time frame a competing 8-bit standard known as EBCDIC was developed by IBM but it it never progressed beyond the mainframe world.

8-bit codes

With CPUs converging on the 8-bit byte as the basic size of addressable memory is was convenient to map one character to one byte (so the in C the default size of type char is 8 bits). In C a string [of characters] just became an array of type char with a null (value 0) terminator. The is a direct correspondance between the string index and the array index so the sixth character in the string is str[5] (C arrays index from 0). With dynamic memory a block of memory can be allocated for the string from heap (using malloc) and assigned to a char* pointer. Now characters could be indexed as *(ptr + index). String overflow problems (where the string is longer than the memory allocated to it) continues to plague many pieces of code up to today.

The move to using 8-bit characters allowed an additional 127 symbols to be encoded. With the rise of digital technology outside the english speaking world it became necessary to include additional characters, either latin characters with accents, for European languages, or entirely new alphabets, for example Cyrillic, Greek, Hebrew or Arabic, became a requirement.

For Chinese, Japanese and Korean (CJK) the alphabets have more symbols than can fit in 8 bits so additional coding methods were required.

The issue around encoding additional characters within the 255 possible codes is there wasn't enough space for everything so the result was multiple (incompatible) code pages.

ISO-8859

The most widely adopted standard for character encodings was the ISO-8859 standard, ratified in 1988. The most widely used was ISO8859-1, which covered most of latin based European languages but not central or eastern Europe. It was updated to ISO8859-15 with the introduction of the Euro symbol.

Unicode

As software became more international it clear that maintaining multiple 8-bit code pages was unsupportable and the inability to mix multiple alphabets in a single piece of text without jumping through multiple hoops was limiting digital technology outside the latin based world.

Starting in 1988 a new ``universal'' character set was based on using a 16-bit code page, called Unicode 88. In 1996 as mechanism was added to extend the code space beyond 16 bits.

Unicode is more than just an encoding, it describes details such as layout (left to right or right to left, or top to bottom), ordering for sorts (called collation) and various typographical features (such as non breaking space).

UTF

The move away from 8-bit code pages had a profound change on string handling in programming languages and how to encode unicode text as binary. The most popular method is use UTF-8 (Unicode Transformation Format). With UTF-8 a unicode character can be anywhere between one and four bytes long.

This encoding is non trivial but has one critical feature, which is a that a UTF-8 string behaves like a normal 8-bit null terminated string provided the code isn't interested in individual characters. Additionally a 7-bit ASCII string is identical to its UTF-8 counterpart.

This allows many older programs to pass UTF-8 encoded strings without modification, so long as their were not doing per character manipulation. This smoothed the adoption of Unicode considerably. The downside of UTF-8 is the character index may longer be the same as the byte index, so to index a character the string needs to be parsed from the start each time. One common solution is to convert the UTF-8 string to either 16-bit or 32-bit characters internally, allowing character to array indexing to match.

Printing Technology

Early printing was based around teletype devices, often modified typewriters connected to telex lines. Many of these typed out the characters onto paper strip. Dedicated printers later emerged, either as pinwheel printers or line printers. In both cases the a physical character imprint was made through an ink ribbon.

While at high end of printing phototypesetters replaced hand typesetting at the other end dot matrix printers replaced pinwheel printers. These allowed the printing of raster (bitmap) images, enabling text to be printed in any font available to the printing software. However they were generally slow, noisy and poor quality.

With the advent of desktop laser printers in the 1980s (initially at 300dpi) high quality printing became accessible. During the 1990s laser printers continued to come down in price and resolution doubled to 600dpi then 1200dpi while inkjets replaced dot matrix printers at the low end, with similar resolution.

Fonts

A font is essentially a collection of related glyphs. They are many and varied but can be arranged by various features.

Monospace With a monospace font every glyph has the same width (assuming horizontal scripts), include the ``space'' character, allowing easy formatting of tables, but harder to read prose. Most terminal fonts are monospace.

Proportional spacing The converse of monospace. Almost all printed (or print like) material uses proportional spaced fonts.

Ultimately all fonts need to be converted to into pixels to be displayed or printed. Original most GUIs came with only bitmap fonts, often at low resolution (with 72dpi or 100dpi). Fonts were originally hand designed then cast into metal. Later these designs were converted into mathematical curves such as bezier curves or splines. The process of converting these mathematical descriptions into pixels is called rasterisation.

There are a number of font formats. These are effectively database files with various tables, only one is the actual glyph description.

Page Description Languages

Page Description Languages are generally used to tell a printer what to print on a page. They vary considerable but the two most common are HP PCL and Adobe Postscript. Adobe PDF is another PDL but most printers do not handle PDFs directly.

PCL is based around escape sequences. These are commands prefixed by the ESCAPE character such as move to, change font, eject page, etc. PCL has developed for inkjets at the time when commands directly corresponded to the movement of the print head and paper roller.

Postscript was designed for laser printers, where the page was first rasterized by the printer into a bitmap before before printing it to paper. It is a full Turing complete language based on forth and could even be used to do computation. One of the major limitations of postscript is its handling of unicode characters.

One feature of early laser printers (such as Apple's Laserwriter) was that the there were standard fonts embedded in the printer but these were not available to the OS. To get things to work low resolution bitmaps fonts were distributed and the font metrics (but not the actual outlines) were made publically available.

These files, known as Adobe Font Metrics (AFM) files, allowed programs such as Pagemaker to accurately layout the page without requiring the full font. Adobe has a number of different font formats but the most common is the Postscript Type1 font.

Due to Adobe's restricted licensing Apple created TrueType to allow MacOS to support outline fonts. Apple licensed the technology to Microsoft for use in Windows.

TrueType was later extended to become OpenType. OpenType added considerable additional configuration for international (ie non latin) scripts.

Typesetting

Typesetting is the process of laying out text onto a page. The process of breaking text into pages is called pagination. Text is collected into paragraphs. The paragraph may be justified (left, right, center or fully jstified). The typesetting may attempt to hypenate a word to improve the balance (in terms of white space) of the paragraph.

Each paragraph is then layouted out onto the page, with rules concerning when a paragraph can be broken (over two pages) or must not go over a page break, in which additional white space needs to be added between the existing paragraphs. In addition page headers and footers may be added and finally watermarking.

Markup

Markup are instructions to a typesetting to layout a piece of text in a particular fashion. This could be the font, size, justification or pagination breaking. These typesetting primitives are similar to printer escape sequences and directly control the final output.

Markup was laster extended to more abstract concepts, such as header text, body text, quotations, etc. This markup was expanded to multiple primitives during the typesetting process.

With computer typesetting it became necessary to manage documents as a collection, giving rise to Document Management Systems (DMS). A form of markup evolved to become SGML (Standardized Generalized Markup Language). This allowed custom tags while enforcing document structure through Document Type Definitions.

HTML is an application of SGML (as it has a specific DTD) but much of the standardization was retrospective rather than planned in advance. In HTML (and SGML) markup is done via the use of elements, such as

,

,

,

etc. These elements could have attributes such as color, fontsize, etc. This was problematic as every element required formatting attributes to be set for consistent output. This was resolved by the use the style sheets.

RUNOFF and ROFF

One of the earliest software typesetting programs was RUNOFF, written in 1964. This was then ported to Unix around 1970 by Ken Thompson as roff and then re-implemented in 1971 for the PDP-11. It was then extended to become ``newer'' roff, or nroff. Later it was extended to ``typesetting'' roff for a CAT phototypesetter, becoming troff. At this point nroff became troff with reduced features, as it only needed to output to a terminal or line printer.

nroff (and troff) are still used today for formatting the Unix man pages, generally using the GNU version of roff, groff. groff has extended the original Unix nroff/troff to support UTF, HTML, postscript and PDF output and other features. However in general its use is fairly limited.

TeX

Donald Knuth, unhappy with the quality of the typesetting for his books on computer science, so in 1978 we wrote and released TeX and Metafont, for typesettings and font rasterizing respectively.

Over the years a number of official version of TeX have been released as well as many derivitives. The source code for TeX and MF are freely available to distribute and modify, with the one restriction that the name must be changed. And example is the original font family designed by Knuth is called Computer Modern, so when additional accented characters were added for european languages, the result was called Latin Modern.

A number of extensions to TeX are currently maintained and a considerable amount of work has gone into supporting non latin scripts and non left to right layout.

Because TeX was designed from the beginning for high quality output of mathematical works TeX is the defacto standard for scientific publishing.

Stylesheets and Macro packages

In TeX and roff documents styling can be changed by using different macro packages, while HTML uses Cascading Style Sheets and MS Word as styles and stylesheets. In each case these are designed to change the look of the output without having to modify the original document (or minimal changes to import a different macro package).

While browsers are designed primarily to render their output to the screen rather than as pages they all function in a similar fashion when converting the input text into glyphs.

Shaping

The process of modifying glyphs based on their neighbouring glyphs is called shaping. With the latin scribe this is fairly limited. The first is when multiply characters are combined into a single glyph, such as tt, ff, fi, ffi. This is known as a ligature. As not all fonts contains specific ligatures this is font specific. The other process is kerning, where space is added or removed from between two characters so that they flow together better. An example is AV. Kerning is font specific, with an effort to minimize whitespace within a word without individual glyphs overlapping.

With many other scripts (for example Devanagari or Arabic) the resulting output is very different from the order of the input text. This can either be done by including instructions embedded in the font files, especially OpenType fonts, or using libraries such as harfbuzz. Layout libraries such as pango allow for multiple shaping engines.