

Plain text

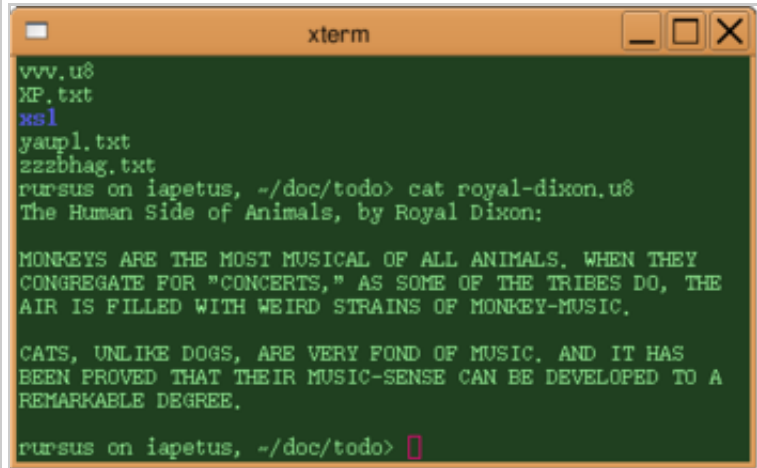
From Wikipedia, the free encyclopedia

In computing, **plain text** is the contents of an ordinary sequential file readable as textual material without much processing, usually opposed to formatted text and to "binary files" in which some portions must be interpreted as binary objects (encoded integers, real numbers, images, etc.).

The encoding has traditionally been either ASCII, one of its many derivatives such as ISO/IEC 646 etc., or sometimes EBCDIC. Unicode-based encodings such as UTF-8 and UTF-16 are gradually replacing the older ASCII derivatives limited to 7 or 8 bit codes.

Contents

- 1 Plain text and rich text
- 2 Plain text, the Unicode definition
- 3 Usage
- 4 Encoding
 - 4.1 Character encodings
 - 4.2 Control codes
- 5 See also
- 6 References

A screenshot of an xterm window with a green background and orange title bar. The window title is 'xterm'. The terminal shows a list of files: 'vvv.u8', 'XP.txt', 'xsl', 'yatpl.txt', and 'zzzbhag.txt'. Below this, the command 'rursus on iapetus, ~/doc/todo> cat royal-dixon.u8' is entered, and its output is displayed: 'The Human Side of Animals, by Royal Dixon:' followed by two paragraphs of text about monkeys and cats. The prompt 'rursus on iapetus, ~/doc/todo>' is shown again at the bottom with a red cursor.

Text file of The Human Side of Animals by Royal Dixon, displayed by the command `cat` in an xterm window

Plain text and rich text

Files that contain markup or other meta-data are generally considered plain-text, as long as the entirety remains in directly human-readable form (as in HTML, XML, and so on (as Coombs, Renear, and DeRose argue,^[1] punctuation is itself markup)). The use of plain text rather than bit-streams to express markup, enables files to survive much better "in the wild", in part by making them largely immune to computer architecture incompatibilities.

According to The Unicode Standard,

- «*Plain text* is a pure sequence of character codes; plain Unicode-encoded text is therefore a sequence of Unicode character codes.»
- *styled text*, also known as *rich text*, is any text representation containing plain text completed by information such as a language identifier, font size, color, hypertext links.^[2]

For instance, Rich text such as SGML, RTF, HTML, XML, and TEX relies on plain text. Wiki technology is another such example.

According to The Unicode Standard, plain text has two main properties in regard to rich text:

- «plain text is the underlying content stream to which formatting can be applied.»
- «Plain text is public, standardized, and universally readable.»^[3]

Plain text, the Unicode definition

- «Plain text represents the basic, interchangeable content of text.»
- «Plain text represents character content only, not its appearance. »
- «It can be displayed in a variety of ways and requires a rendering process to make it visible with a particular appearance.»
- «If the same plain text sequence is given to disparate rendering processes, there is no expectation that rendered text in each instance should have the same appearance. »
- «Instead, the disparate rendering processes are simply required to make the text legible according to the intended reading. »
- «This legibility criterion constrains the range of possible appearances. »
- «The relationship between appearance and content of plain text may be summarized as follows: Plain text must contain enough information to permit the text to be rendered legibly, and nothing more.»
- «The Unicode Standard encodes plain text.»
- «The distinction between plain text and other forms of data in the same data stream is the function of a higher-level protocol and is not specified by the Unicode Standard itself.»^[4]

More formally, the fundamental distinction of "plain text" is that no information would be lost if you went through and translated the file to a completely different character encoding, or translated it to *no* encoding by just printing it out generically (provided the printer has a good enough font that you can correctly distinguish all the characters!). No information is conveyed by the fact that an "A" in the printout was originally stored as a byte with value 65 (as it would be in ASCII), or with value 193 (as in EBCDIC); and it certainly wasn't meant to express half of the bits of an integer.^[citation needed]

Usage

The purpose of using ***plain text*** today is primarily independence from programs that require their very own special encoding or formatting, and from computer architecture issues such as byte order, etc. Plain text files can be opened, read, and edited with countless generic text editors and utilities. Examples include Notepad (Windows), edit (DOS), ed, emacs, vi, vim, Gedit or nano (Unix, Linux), SimpleText (Mac OS), or TextEdit (Mac OS X).

Many other computer programs are also capable of processing or creating plain text, such as countless commands in DOS, Windows, MacOS, and Unix and its kin; as well as web browsers (a few browsers such as Lynx and the Line Mode Browser produce only plain text for display).

Plain text files are almost universal in programming; a source code file containing instructions in a programming language is almost always a plain text file. Plain text is also commonly used for configuration files, which are read for saved settings at the startup of a program, and for much e-mail.

Encoding

Character encodings

Main article: Character encoding

Before the early 1960s, computers were mainly used for number-crunching rather than for text, and memory was extremely expensive. Computers often allocated only 6 bits for each character, permitting only 64 characters—assigning codes for A-Z, a-z, and 0-9 would leave only 2 codes: nowhere near enough. Most computers opted not to support lower-case letters. Thus, early text projects such as Roberto Busa's Index Thomisticus, the Brown Corpus, and others had to resort to conventions such as keying an asterisk preceding letters actually intended to be upper-case.

Fred Brooks of IBM argued strongly for going to 8-bit bytes, because someday people might want to process text; and won. Although IBM used EBCDIC, most text from then on came to be encoded in ASCII, using values from 0 to 31 for (non-printing) control characters, and values from 32 to 127 for graphic characters such as letters, digits, and punctuation. Most machines stored characters in 8 bits rather than 7, ignoring the remaining bit or using it as a checksum).

The near-ubiquity of ASCII was a great help, but failed to address international and linguistic concerns. The dollar-sign ("\$\$") was not so useful in England, and the accented characters used in Spanish, French, German, and many other languages were entirely unavailable in ASCII (not to mention characters used in Greek, Russian, and most Eastern languages). Many individuals, companies, and countries defined extra characters as needed—often re-assigning control characters, or using value in the range from 128 to 255. Using values above 128 conflicts with using the 8th bit as a checksum, but the checksum usage gradually died out.

These additional characters were encoded differently in different countries, making texts impossible to decode without figuring out the originator's rules. For instance, a browser might display ¬A rather than ` if it tried to interpret one character set as another. The International Organisation for Standardisation (ISO) eventually developed several code pages under ISO 8859, to accommodate various languages. The first of these (ISO 8859-1) is also known as "Latin-1", and covers the needs of most (not all) European languages that use Latin-based characters (there was not quite enough room to cover them all). ISO 2022 then provided conventions for "switching" between different character sets in mid-file. Many other organisations developed variations on these, and for many years Windows and Macintosh computers used incompatible variations.

The text-encoding situation became more and more complex, leading to efforts by ISO and by the Unicode Consortium to develop a single, unified character encoding that could cover all known (or at least all currently known) languages. After some conflict,^[*citation needed*] these efforts were unified. Unicode currently allows for 1,114,112 code values, and assigns codes covering nearly all modern text writing systems, as well as many historical ones and for many non-linguistic characters such as printer's dingbats, mathematical symbols, etc.

Text is considered plain-text regardless of its encoding. To properly understand or process it the recipient must know (or be able to figure out) what encoding was used; however, they need not know anything about the computer architecture that was used, or about the binary structures defined by whatever program (if any) created the data.

Control codes

Main article: Newline

The ASCII codes before `SPACE` (`= 32 = 20H`) are not intended as displayable characters, but instead as control characters. They are used for diverse interpreted meanings. For example, the code `NULL` (`= 0`, sometimes denoted `Ctrl-@`) is used as string end markers in the programming language C and successors. Most troublesome of these are the codes `LF` (`= LINE FEED = 10 = 0AH`) and `CR` (`= CARRIAGE RETURN = 13 = 0DH`). Windows and OS/2 require the sequence `CR, LF` to represent a newline, while Unix and relatives use just the `LF`, and Classic Mac OS (but not Mac OS X) uses just the code `CR`. This was once a slight problem when transferring files between Windows and Unix systems, but today most computer programs treat this seamlessly.

In 8-bit character sets such as Latin-1 and the other ISO 8859 sets, the first 32 characters of the "upper half" (128 to 159) are also control codes, known as the "C1 set" as opposed to the "C0" set just described. However, the commonplace Windows character set called code page 1252 assigns printing characters to these code points (other than this, cp1252 is the same as Latin-1). It is not uncommon that Web servers identify a document as being in Latin-1, when in fact it is in code page 1252, and uses characters in the C1 set as graphics. This may or may not lead to unexpected results.

See also

- Plaintext, most commonly used in a cryptographic context
- Cleartext usually refers to lack of protection from eavesdropping
- E-text
- MIME Content-type
- File format
- Binary file
- Text file
- Editor wars
- Source code

References

- ↑ Coombs, James H.; Renear, Allen H.; DeRose, Steven J. (November 1987). "Markup systems and the future of scholarly text processing" (<http://xml.coverpages.org/coombs.html>). *Communications of the ACM* (ACM) **30** (11): 933–947. doi:10.1145/32206.32209 (<http://dx.doi.org/10.1145%2F32206.32209>).
- ↑ The Unicode Standard, version 6.1, General Structure, page 14 (<http://www.unicode.org/versions/Unicode6.1.0/>)
- ↑ The Unicode Standard, version 6.1, General Structure, page 14 (<http://www.unicode.org/versions/Unicode6.1.0/>)
- ↑ The Unicode Standard, version 6.1, General Structure, page 15 (<http://www.unicode.org/versions/Unicode6.1.0/>)

Retrieved from "http://en.wikipedia.org/w/index.php?title=Plain_text&oldid=548508306"

Categories: Computer file formats

-
- This page was last modified on 3 April 2013 at 16:26.
 - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply.

By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#).

Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.