Unicode is a computing standard for the encoding, it doesn’t depend on the platform, language or program.

Nowadays Unicode is the most popular encoding standard in the Internet because it provides encoding of various types of symbols, including different language systems.

Those symbols can be:

* Hieroglyphs;
* Math’s symbols;
* Any alphabetical symbols (e.g. Greek, Latin, Cyrillic, etc.);
* Musical notes.

As far as Unicode is not a coding system, but a standard on encoding only, it doesn’t determine the way of how the symbols could be stored on the HDD or how they can be transferred via the network. The only use of the Unicode is determination of the connection between the symbol and the number.

The format used in coding symbols is determined by Unicode Transformation Formats. In this article two types of formats are reviewed: UTF-8 and UTF-16.

Starting from the definition of UTF: as it was said, UTF is Unicode Transformation Format and it is the way symbols are changed in coding system.

UTF-8 is the most popular 8-bit format. At provides compact storage and transfer of the symbols of the Unicode. This format uses variable number of bytes (from 1 to 4). Also it has such an advantage as compatibility with ASCII coding.

The UTF-8 encoding has 3 steps:

1. Determine the amount of the octets (bytes) essential for the encoding of the numeric;
2. Determine big-endian of the 1st octet according to the necessary amount of the octets (determined in Step 1);
3. Set significant bits of the octet(-s) according to the symbol number in Unicode in binary number.

More details you can see in the example with the symbol $:

According to the Unicode, this symbol has 36 number.

In binary system 36 is recorded as 100100.

As far as 36 suits the 8 bits, 1 octet is enough.

Then big-endian is set.

In case of the only octet 0 is set as big-endian in order to number 36 could suit 8 bits. As a result, there is 8-bit mold: 0xxxxxxx.

The final step is determination of the significant bits according to the symbol number in Unicode in binary system. The determination goes from the right end to the left. Free bits are filled with 0 (zero).

According to these steps the $ symbols in UTF-8 looks like 00100100.

In UTF-8 texts in Latin are the most compact because all Latin symbols (without diacritics), numbers and most popular punctuation marks are encoded in 1 byte.

UTF-16 is also Transformation Format. It is presented as the sequence of 16-bit units. According to this format every symbol can be reflected as 1 or 2 units.

In comparison with compact format UTF-8, UTF-16 ecnodes symbols in 2-byte units.

Here are advantages and disadvantages of each format.

UTF-8:

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Main ASCII symbols (numbers, latin symbols without diacritics, punctuation marks) are equal to 1 byte. It is similar to US-ASCII and thus there is backward compatibility between UTF-8 and ASCII. | Many symbols have different length and this fact slows the code identification down. |
| There are no zero-bytes; it allows using lines with zero symbol at the end. It also provides backward compatibility. | Even if the byte order is not obligatory, UTF-8 has a specific feature (byte order mark) which is used for notification about the text encoding in UTF-8. It can violate the compatibility with ASCII software. |
| UTF-8 does not depend on byte order, therefore big endian and small endian are not a problem. |  |

UTF-16:

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| The BMP (basic multilingual plane) symbols like Latin, Cyrillic, most of Chinese and Japanese languages can be encoded in 2 bytes. This fact fastens indexing and calculation of the code points in case if there are no special symbols in the text. | Lots of zero-bytes in US-ASCII lines, it means that there is no line with zero symbol at the end and lots of lost storage. |
| If there are special symbols in the text, they will be encoded in 16-bit couples. It means that the common length still can be divided by 2 and allows using 16-bit symbol type as simple line component. | Usage of this format as the format with fixed length is popular in usual cases (it means used in USA, EU countries, countries with Cyrillic alphabets, Israel, Arabian countries, etc.). But in others UTF-16 causes errors and violates the maintenance. |
|  | As far as UTF-16 is based on variable length, thus calculation or indexing of code points take more time. |

In conclusion: UTF-16 is better choice for storage performance. UTF-8 is more convenient for text files and network protocols.