Unicode and MultiLingual Computing

Created in Master PDF Editor

Contents

Created in Master PDF Editor

What is Unicode

an international encoding standard for use with different languages and scripts, by which each letter, digit, or symbol is assigned a unique numeric value that applies across different platforms and programs. - Source (Definition provided by Google)

Unicode provides a unique number for every character,

no matter what the platform,

no matter what the program,

no matter what the language.

- Source (http://unicode.org/standard/WhatIsUnicode.html)

Created in Master PDF Editor

Who Uses Unicode

- The Unicode standard has been adopted by such industry leaders as Apple, Google, HP, IBM, JustSystems, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and many others.
- Unicode is required by modern standards such as XML, Java, ECMAScript(JavaScript), LDAP, CORBA 3.0, WML etc. and is the official way to implement ISO/IEC 10646.
- It is supported in many operating systems, all modern browsers, and many other products.
- The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends.

Created in Master PDF Editor

Benefits of Using Unicode

- Incorporating Unicode into client-server or multi-tiered applications and websites offers significant cost savings over the use of legacy character sets.
- Unicode enables a single software product or a single website to be targeted across multiple platforms, languages and countries without re-engineering.
- It allows data to be transported through many different systems without corruption.

Created in Master PDF Editor

About the Unicode Consortium (www.unicode.org)

- The Unicode Consortium was founded to develop, extend and promote use of the Unicode Standard, which specifies the representation of text in modern software products and standards.
- The Consortium is a non-profit, charitable organization.
- The membership of the Consortium represents a broad spectrum of corporations and organizations in the computer and information processing industry.

Created in Master PDF Editor

Encoding Basics

- Please go through the article - “The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)” by Joel Spolsky, co-founder of Trello and Fog Creek Software, and CEO of Stack Overflow.
- FAQ from the Official Unicode website, http://www.unicode.org/faq//utf_bom.html
- Why do we need Unicode?
  - Lets say Joe Average is a software developer. He insists that he will only ever need English, and as such only wants to use ASCII. This might be fine for Joe the *user,* but this is not fine for Joe the *software developer*. Approximately, half the world uses the non-Latin characters and using ASCII is arguably inconsiderate to these people, and on top of that, he is closing off his software to a large and growing economy.
  - Therefore, an encompassing character set including *all* languages is needed. Thus came Unicode. It assigns every character a unique number called a **code point**.

Created in Master PDF Editor

Encoding Basics

- One advantage of Unicode over other possible sets is that the first 256 code points are identical to ISO-8859-1, https://en.wikipedia.org/wiki/ISO/IEC_8859-1, and hence also ASCII.
- In addition, the vast majority of commonly used characters are represented by only two bytes, in a region called the Basic Multilingual Plane (BMP), https://en.wikipedia.org/wiki/Plane_%28Unicode%29#Basic_Multilingual_Plane
- Memory considerations
  - UTF-8:
    - 1 byte: Standard ASCII
    - 2 bytes: Arabic, Hebrew, most European scripts (most notably excluding Georgian, https://en.wikipedia.org/wiki/Georgian_scripts)
    - 3 bytes: BMP
    - 4 bytes: All Unicode characters
  - UTF-16:
    - 2 bytes: BMP
    - 4 bytes: All Unicode Characters

Created in Master PDF Editor

Encoding Basics

- It’s worth mentioning that characters not in the BMP include ancient scripts, mathematical symbols, musical symbols, and rarer Chinese/Japanese/Korean (CJK) characters, http://www.unicode.org/faq/han_cjk.html
- If you’ll be working mostly with ASCII characters, then UTF-8 is certainly more memory efficient.
- However, if you are working mostly with non-European scripts, using UTF-8 could be up to 1.5 times less memory efficient than UTF-16.
- When dealing with large amounts of text, such as large web-pages or lengthy word documents, this could impact performance.
- **UTF-8:** For the standard ASCII (0-127) characters, the UTF-8 codes are identical. This makes UTF-8 ideal if backwards compatibility is required with existing ASCII text. Other characters require anywhere from 2-4 bytes. This is done by reserving some bits in each of these bytes to indicate that it is part of a multi-byte character. In particular, the first bit of each byte is 1 to avoid clashing with the ASCII characters.

Created in Master PDF Editor

Encoding Basics

- **UTF-16:** For valid BMP characters, the UTF-16 representation is simply its code point. However, for non-BMP characters UTF-16 introduces surrogate pairs. In this case a combination of two two-byte portions map to a non-BMP character. These two-byte portions come from the BMP numeric range, but are guaranteed by the Unicode standard to be invalid as BMP characters. In addition, since UTF-16 has two bytes as its basic unit, it is affected by endianness. To compensate, a reserved byte order mark can be placed at the beginning of a data stream which indicates endianness. Thus, if you are reading UTF-16 input, and no endianness is specified, you must check for this.
- As can be seen, UTF-8 and UTF-16 are nowhere near compatible with each other. So if you're doing I/O, make sure you know which encoding you are using! For further details on these encodings, please see the UTF FAQ, http://www.unicode.org/faq/utf_bom.html

Created in Master PDF Editor

Encoding Basics

- **Practical programming considerations**
  - **Character and String data types:** How are they encoded in the programming language? If they are raw bytes, the minute you try to output non-ASCII characters, you may run into a few problems. Also, even if the character type is based on a UTF, that doesn't mean the strings are proper UTF. They may allow byte sequences that are illegal. Generally, you'll have to use a library that supports UTF, such as ICU for C, C++ and Java. In any case, if you want to input/output something other than the default encoding, you will have to convert it first.
  - **Recommended/default/dominant encodings:** When given a choice of which UTF to use, it is usually best to follow recommended standards for the environment you are working in. For example, UTF-8 is dominant on the web, and since HTML5, it has been the recommended encoding. Conversely, both .NET and Java environments are founded on a UTF-16 character type. Confusingly (and incorrectly), references are often made to the "Unicode encoding", which usually refers to the dominant UTF encoding in a given environment.

Created in Master PDF Editor

Encoding Basics

- **Practical programming considerations**
  - **Library support:** What encodings are the libraries you are using support? Do they support the corner cases? Since necessity is the mother of invention, UTF-8 libraries will generally support 4-byte characters properly, since 1, 2, and even 3 byte characters can occur frequently. However, not all purported UTF-16 libraries support surrogate pairs properly since they occur very rarely.
  - **Counting characters:** There exist combining characters in Unicode. For example the code point U+006E (n), and U+0303 (a combining tilde) forms ñ, but the code point U+00F1 forms ñ. They should look identical, but a simple counting algorithm will return 2 for the first example, 1 for the latter. This isn't necessarily wrong, but may not be the desired outcome either.
  - **Comparing for equality:** A, А, and Α look the same, but they're Latin, Cyrillic, and Greek respectively. You also have cases like C and Ⅽ, one is a letter, the other a Roman numeral. In addition, we have the combining characters to consider as well. For more info see Duplicate characters in Unicode, https://en.wikipedia.org/wiki/Duplicate_characters_in_Unicode.

Created in Master PDF Editor

Encoding Basics

- **Practical programming considerations**
  - **Surrogate pairs:** These come up often enough on. So, here are some example links
  - Getting string length, http://stackoverflow.com/questions/12907022/python-getting-correct-string-length-when-it-contains-surrogate-pairs
  - Removing surrogate pairs, http://stackoverflow.com/questions/12867000/how-to-remove-surrogate-characters-in-java
  - Palindrome checking http://stackoverflow.com/questions/12340930/string-is-palindrome-or-not/12341017#12341017

Created in Master PDF Editor