Programming in Graphical Environment

Internationalization

Paweł Aszklar
pawel.aszklar@pw.edu.pl

Faculty of Mathematics and Information Science
Warsaw Univeristy of Technology

Warsaw 2024

Internationalization

Internationalization A.k.a. *Globalization*, process of designing and implementing software so that it can be easily adapted to other languages and cultures (*i18n*, *g11n*).
Includes:

- Separation of formatting of textual and other data from main logic,
- UI design adaptable to changing text length, layout etc.,
- Ability to handle international scripts,
- Preference for images/icons with or instead of text to convey meaning.

Localization Process of adapting software to particular language and/or culture (*l10n*)

`https://www.w3.org/International`
`https://www.unicode.org/`

Glossary

Character No single definition:

- Smallest component of a written language (refers to semantic meaning, abstract shape rather than specific rendering of that shape i.e. *glyph*)
- Basic unit of particular character set

Abstract character Smallest unit of information for organization, control and representation of textual data (letters, digits, diacritic marks, control characters, …)

User perceived character What a person thinks of as a character in their script

Code point Numerical value, might correspond to a character (but doesn't have to)

Coded character set Mapping of abstract characters to code points (a.k.a. *character set*, *character encoding*)

Text encoding Method of storing and/or transmitting code points

Glossary

Grapheme Minimally distinctive unit of writing for a given writing system. Characters are not distinct graphemes if their substitution doesn't change meaning. Grapheme can be represented using multiple code points, or vice versa.

Grapheme cluser Horizontally segmentable unit of text ($\approx$ user perceived character)

Glyph Particular image representing piece of textural data. *Code point* might be rendered using multiple glyphs, or multiple grapheme clusters might be represented by single glyph (e.g. ligatures). Glyphs can have variants (i.e. different image representation of the same data)

ASCII

- No such thing as plain text
- As with any data, values meaningless without proper interpretation
- Many text encodings existed in the past, eg. EBCIDIC
- ASCII — one of the more popular early computer text encodings
  - Standards: ISO/IEC 646, ECMA-6, ANSI X3.4
  - 7-bit encoding
  - 128 code-points

Code Pages

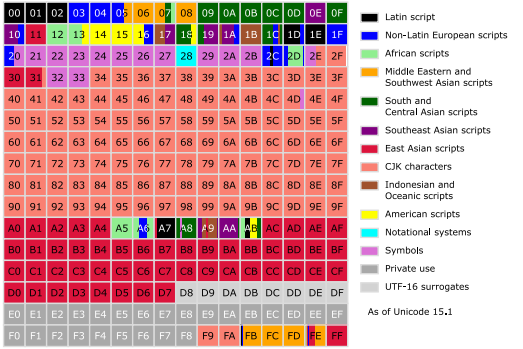
- Minimum data granularity on most computers is 8-bits
- Extra bit used to encode more characters
- Different approaches often called code pages: ISO-8859-x, Windows-125x, KOI8-R …
- Various degrees of standardization: ISO/IEC 8859
- Single-byte code pages
  - Additional 128 code points
  - Sufficient for many languages, but also many exceptions: Eastern-Asian scripts, …
- Double- or multi-byte character sets
  - Usually First byte with highest bit set interpreted together with the next one
  - Over 65 thousand possible code-points
  - GBK, Big5
  - Others use shift states (code pointe sequences switch between character sets): ISO/IEC 2022
  - No way to determine if randomly selected byte is stand-alone, first or second byte of a pair
- Mojibake - incorrect text rendering if wrong code page is used

Universal Character Set

- Two independent attempts to define unified international character set, eventually merged
  - ISO 10646
  - The Unicode Standard
- Current ISO/IEC 10646:2021 defines ($\approx$ Unicode 14.0):
  - Code point space: `0-0x10FFFF` of 1 114 112 possible code-points divided into 17 planes
  - Some code points are explicitly disallowed resulting in 1 111 998 usable values
  - Extends ASCII code space
  - Coded character set: currently maps 144 697 characters
  - Several text encodings: UCS2 (deprecated), UCS4/UTF-32, UTF-8, UTF-16
- Unicode Standard (Currently v. 15.1):
  - More frequently updated (now 149 813 assigned code points)
  - Standardizes additional rules for: collation (string ordering), normalization, bi-directional layout rendering
  - Those rules require to assign additional properties and meaning to each character

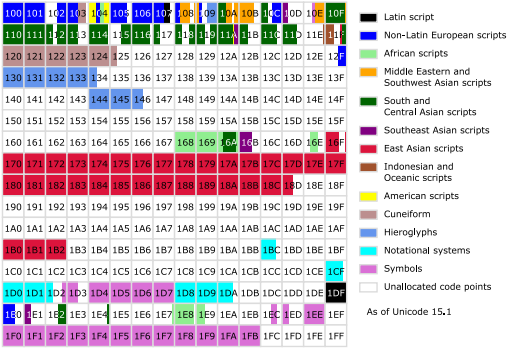
Code Space Allocation

0 – Basic Multilingual Plane



By Drmccreedy - Self-made using this perl script and information from Unicode., Public Domain

1 – Supplementary Multilingual Plane



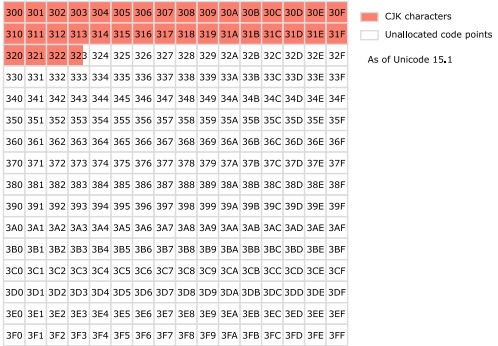
By Drmccreedy - Own work
(Original text: Self-made using this perl script and information from Unicode.), Public Domain

Code Space Allocation

2 – Supplementary Ideographic Plane

By Drmccreedy - Self-made using this perl script and information from Unicode., CC BY-SA 4.0

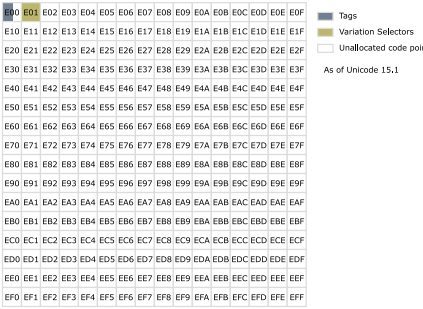
3 – Tertiary Ideographic Plane



By Drmccreedy - Self-made using this perl script and information from Unicode., CC BY-SA 4.0

Code Space Allocation

14 – Supplementary Special-purpose Plane



By Drmccreedy - Self-made using this perl script and information from Unicode., CC BY-SA 4.0

4-13 – Unassigned Planes

15-16 – Private Use Area Planes

Universal Character Set

Code point properties:

- Basic type (graphic, format and control types are collectively known as Characters):
  - Graphic – have visual representation (have a visible glyph or represent visible space)
  - Format – modify appearance of other characters, but have no visual of their own
  - Control – 65 control codes for ISO/IEC 6429 compatibility (ANSI escape codes)
  - Private-use – Will never be assigned specific interpretation
  - Surrogate, non-character – disallowed code points
  - Reserved – code points not yet assigned

Universal Character Set

Code point properties:

- Major and minor category (all graphic type except last two):
  - Letters: uppercase, lowercase, titlecase (e.g. ligatures of upper- and lowercase letters), modifiers (diacritics), other (ideographs, unicase alphabets)
  - Marks: spacing, non-spacing, enclosing
  - Numbers: decimal, letter (e.g. Roman numerals), other (e.g. fractions, superscripts, subscripts)
  - Punctuation: connectors (e.g. ”_”), dashes, opening, closing brackets, quotes, other
  - Symbols: maths, currency, other
  - Separators: white-space characters (graphic type), line, paragraph separators (format type)
  - Other: control, format, surrogate, private-use, non-character, reserved

Universal Character Set

Code point properties:

- Name
- Canonical Combining Class
- Bidirectional Class
- Bidirectional Mirroring
- Decomposition type and mapping
- Simple uppercase, lowercase, titlecase mapping

Normalization

- Not all grapheme clusters have individual code points
- Some require composition: e.g. base letter and few diacritic combination marks
- However, some most common compositions assigned to single code points
- Some ligatures (e.g. ”fi”) also have their own code points even though they represent multiple graphemes

Normalization

Equivalent text can be represented in multiple ways

- Canonical equivalence, e.g. Å:
  - Latin Capital Letter A with Ring Above U+00C5
  - Angstrom Sign U+212B
  - Latin Capital Letter A U+0041 + Combining Ring Above U+030A
- Compatible equivalence
  - ¼ (U+00BC) $\rightarrow$ 1⁄4 (U+0031 U+2044 U+0034)
  - x³ (U+0078 U+00B3) $\rightarrow$ x3 (U+0078 U+)
  - ℜ (U+211C) $\rightarrow$ R (U+0052 U+0033)

Normalization

Normalization required for text comparison

- Normalization Forms:
  - NFD Decomposed form, Canonical decomposition splitting into base character and combination marks
  - NFC Composed form, Canonical decomposition followed by canonical composition.
  - NFKC, NFKD Compatibility variants of the above, split ligatures, convert subscripts, superscripts, etc. to decimal numbers, …
- All reorder remaining combination marks: one that go below a character come first
- Canonical forms for strong equality, Compatible forms for string search
- Decomposed forms make it easy to find base character
- May still require case mapping (Uppercase vs lowercase vs titlecase)
- All normalization can cause loss of information

`https://www.unicode.org/reports/tr15/`

Algorithms

Unicode provides standardized algorithms for various text-related problems, e.g.:

- Normalization Forms (mentioned previously)
- Bidirectional Algorithm - character positioning in mixed LTR and RTL flow.
  `https://www.unicode.org/reports/tr9/`
- Collation Algorithm - string ordering
  `https://www.unicode.org/reports/tr10/`
- Line Breaking - `https://www.unicode.org/reports/tr14/`
- Text Segmentation - boundaries between ”user-perceived characters” (grapheme clusters), words, sentences.
  `https://www.unicode.org/reports/tr29/`

These algorithms are reasonable, but can never be perfect.

Encodings

UCS-2 Deprecated. Stores code point values as two bytes. Doesn't encode entire code space.

UTF-32 A.k.a. UCS-4, stores code points as 4-byte values. High overhead for most common characters

UTF-8 A.k.a. one of the greatest programming hacks in history, variable multi-byte encoding

- Code points encoded as multi-byte sequences:

UTF-8 (binary)	Code point (binary)	Range
0xxxxxxx	xxxxxxx	U+0000 - U+007F
110xxxxx 10yyyyyy	xxxxxyyyyyy	U+0080 - U+07FF
1110xxxx 10yyyyyy 10zzzzzz	xxxxyyyyyyzzzzzz	U+0800 - U+FFFF
11110xxx 10yyyyyy 10zzzzzz 10wwwwww	xxxyyyyyyzzzzzzwwwwww	U+10000 - U+10FFFF

- ASCII characters encoded without change (high bit is 0)
- Random byte can be easily determined to be single-byte character, start or continuation of multi-byte sequence
- At most 4 steps needed to find beginning of encoded code point.
- Extension possible up to 2^{31} values, not just 2^{21} Unicode requires

Encodings

UTF-16 Variable length multi-byte encoding

- Code points encoded as one or two 16-bit code units:

UTF-16 (binary)	Code point (binary)	Range
xxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxx	U+0000 - U+FFFF
110110xxxxxxxxxx 110111yyyyyyyyyy	xxxxxxxxxxyyyyyyyyyy + 0x10000	U+10000 - U+10FFFF

- Compatible with UCS-2 (*high* surrogates, U+D800–U+DBFF, i.e. 2-byte code-points starting with 110110 and *low* surrogates, U+DC00–U+DFFF, i.e. 2-byte code-points starting with 110111 are explicitly marked as invalid)
- Susceptible to corruption when transferring data between system of differing endianness
- Zero-width non-breakable space U+FEFF BOM (byte-order mark) used often (inverting bytes produces invalid code-point U+FFFE indicating endianness change)

UTF-7 Not standardized, Bas64 encoded UTF-16

Other Capable of representing full code space: GB 18030, UTF-1, BOCU, SCSU

Locale

- Unicode support simplifies localization, but only for displaying human-generated text.
- (Semi-)Automatic conversions to/from other data types and their textual representation is more difficult.
- Different rules apply not only across languages, but also countries, regions, ethnicities and cultures.
- Set of such rules is called locale
- Unicode Common Locale Data Repository: `https://cldr.unicode.org`

Locale

Non-exhaustive list of locale conventions

- Language Properties
  - Characters used and keyboard layout
  - Rules for punctuation, capitalization, collation, text segmentation.
  - Plural cases, grammatical genders
  - Formatting of lists
  - etc., etc.
- Numbers
  - Numeral System
  - Number formatting and parsing
  - Currency, units
  - Number spelling

Locale

Non-exhaustive list of locale conventions

- Date and Time
  - Calendar, week convention, leap years, leap seconds
  - Date and time formatting in different contexts (short, long, relative, absolute, . . .)
  - Time zones, daylight savings time
- Translations of names:
  - Languages and scripts
  - Countries, regions, territories, cities, . . .
  - Eras, moths, weekdays, day periods, time zones
  - Currencies and other units

End of Internationalization

Thank you for listening! ☺