

Programming in Graphical Environment

Internationalization

Paweł Aszklar
pawel.aszklar@pw.edu.pl

Faculty of Mathematics and Information Science
Warsaw University 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 cluster** 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. EBCDIC
- 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

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_0	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
1_16	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
2_32	SP 0020	! 0021	" 0022	# 0023	\$ 0024	% 0025	& 0026	' 0027	(0028) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3_48	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_64	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_80	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C] 005D	^ 005E	_ 005F
6_96	 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_112	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	 007C	} 007D	~ 007E	DEL 007F

Letter
 Number
 Punctuation
 Symbol
 Other
 undefined
 Changed from 1963

Code Pages

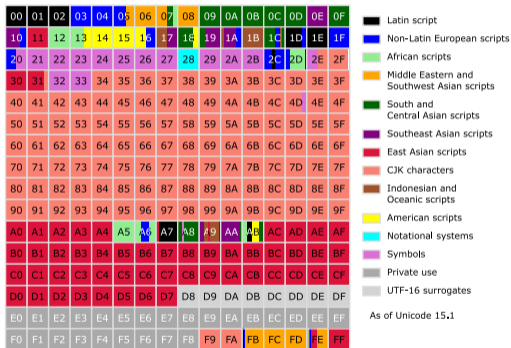
- Minimum data granularity on most computers is 8-bits
- Extra bit used to encode more characters
- Different approaches often called code pages: ISO-8859-x, Windows-125x, KOI8-R ...
- Various degrees of standardization: ISO/IEC 8859
- Single-byte code pages
 - Additional 128 code points
 - Sufficient 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 point 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

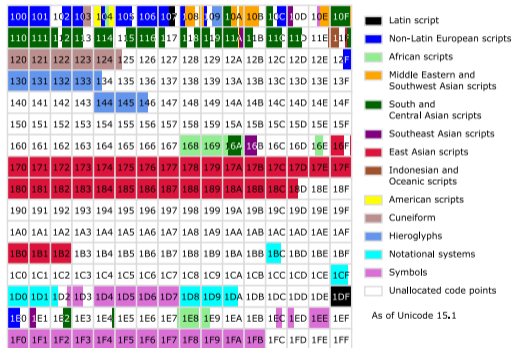
- Two independent attempts to define unified international character set, eventually merged
 - ISO 10646
 - The Unicode Standard
- Current ISO/IEC 10646:2021 defines (\approx Unicode 14.0):
 - Code point space: $0-0x10FFFF$ of 1 114 112 possible code-points divided into 17 planes
 - Some code points 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



1 – Supplementary Multilingual Plane



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

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

Code Space Allocation

2 – Supplementary Ideographic Plane

200	201	202	203	204	205	206	207	208	209	20A	20B	20C	20D	20E	20F
210	211	212	213	214	215	216	217	218	219	21A	21B	21C	21D	21E	21F
220	221	222	223	224	225	226	227	228	229	22A	22B	22C	22D	22E	22F
230	231	232	233	234	235	236	237	238	239	23A	23B	23C	23D	23E	23F
240	241	242	243	244	245	246	247	248	249	24A	24B	24C	24D	24E	24F
250	251	252	253	254	255	256	257	258	259	25A	25B	25C	25D	25E	25F
260	261	262	263	264	265	266	267	268	269	26A	26B	26C	26D	26E	26F
270	271	272	273	274	275	276	277	278	279	27A	27B	27C	27D	27E	27F
280	281	282	283	284	285	286	287	288	289	28A	28B	28C	28D	28E	28F
290	291	292	293	294	295	296	297	298	299	29A	29B	29C	29D	29E	29F
2A0	2A1	2A2	2A3	2A4	2A5	2A6	2A7	2A8	2A9	2AA	2AB	2AC	2AD	2AE	2AF
2B0	2B1	2B2	2B3	2B4	2B5	2B6	2B7	2B8	2B9	2BA	2BB	2BC	2BD	2BE	2BF
2C0	2C1	2C2	2C3	2C4	2C5	2C6	2C7	2C8	2C9	2CA	2CB	2CC	2CD	2CE	2CF
2D0	2D1	2D2	2D3	2D4	2D5	2D6	2D7	2D8	2D9	2DA	2DB	2DC	2DD	2DE	2DF
2E0	2E1	2E2	2E3	2E4	2E5	2E6	2E7	2E8	2E9	2EA	2EB	2EC	2ED	2EE	2EF
2F0	2F1	2F2	2F3	2F4	2F5	2F6	2F7	2F8	2F9	2FA	2FB	2FC	2FD	2FE	2FF

■ CJK characters
 Unallocated code points
 As of Unicode 15.1

3 – Tertiary Ideographic Plane

300	301	302	303	304	305	306	307	308	309	30A	30B	30C	30D	30E	30F
310	311	312	313	314	315	316	317	318	319	31A	31B	31C	31D	31E	31F
320	321	322	323	324	325	326	327	328	329	32A	32B	32C	32D	32E	32F
330	331	332	333	334	335	336	337	338	339	33A	33B	33C	33D	33E	33F
340	341	342	343	344	345	346	347	348	349	34A	34B	34C	34D	34E	34F
350	351	352	353	354	355	356	357	358	359	35A	35B	35C	35D	35E	35F
360	361	362	363	364	365	366	367	368	369	36A	36B	36C	36D	36E	36F
370	371	372	373	374	375	376	377	378	379	37A	37B	37C	37D	37E	37F
380	381	382	383	384	385	386	387	388	389	38A	38B	38C	38D	38E	38F
390	391	392	393	394	395	396	397	398	399	39A	39B	39C	39D	39E	39F
3A0	3A1	3A2	3A3	3A4	3A5	3A6	3A7	3A8	3A9	3AA	3AB	3AC	3AD	3AE	3AF
3B0	3B1	3B2	3B3	3B4	3B5	3B6	3B7	3B8	3B9	3BA	3BB	3BC	3BD	3BE	3BF
3C0	3C1	3C2	3C3	3C4	3C5	3C6	3C7	3C8	3C9	3CA	3CB	3CC	3CD	3CE	3CF
3D0	3D1	3D2	3D3	3D4	3D5	3D6	3D7	3D8	3D9	3DA	3DB	3DC	3DD	3DE	3DF
3E0	3E1	3E2	3E3	3E4	3E5	3E6	3E7	3E8	3E9	3EA	3EB	3EC	3ED	3EE	3EF
3F0	3F1	3F2	3F3	3F4	3F5	3F6	3F7	3F8	3F9	3FA	3FB	3FC	3FD	3FE	3FF

■ CJK characters
 Unallocated code points
 As of Unicode 15.1

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

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

Code Space Allocation

14 – Supplementary Special-purpose Plane

4-13 – Unassigned Planes

15-16 – Private Use Area Planes

E00	E01	E02	E03	E04	E05	E06	E07	E08	E09	E0A	E0B	E0C	E0D	E0E	E0F
E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E1A	E1B	E1C	E1D	E1E	E1F
E20	E21	E22	E23	E24	E25	E26	E27	E28	E29	E2A	E2B	E2C	E2D	E2E	E2F
E30	E31	E32	E33	E34	E35	E36	E37	E38	E39	E3A	E3B	E3C	E3D	E3E	E3F
E40	E41	E42	E43	E44	E45	E46	E47	E48	E49	E4A	E4B	E4C	E4D	E4E	E4F
E50	E51	E52	E53	E54	E55	E56	E57	E58	E59	E5A	E5B	E5C	E5D	E5E	E5F
E60	E61	E62	E63	E64	E65	E66	E67	E68	E69	E6A	E6B	E6C	E6D	E6E	E6F
E70	E71	E72	E73	E74	E75	E76	E77	E78	E79	E7A	E7B	E7C	E7D	E7E	E7F
E80	E81	E82	E83	E84	E85	E86	E87	E88	E89	E8A	E8B	E8C	E8D	E8E	E8F
E90	E91	E92	E93	E94	E95	E96	E97	E98	E99	E9A	E9B	E9C	E9D	E9E	E9F
EA0	EA1	EA2	EA3	EA4	EA5	EA6	EA7	EA8	EA9	EAA	EAB	EAC	EAD	EAE	EAF
EB0	EB1	EB2	EB3	EB4	EB5	EB6	EB7	EB8	EB9	EBA	EBB	EBC	EBD	EBE	EBF
EC0	EC1	EC2	EC3	EC4	EC5	EC6	EC7	EC8	EC9	ECA	ECB	ECC	ECD	ECE	ECF
ED0	ED1	ED2	ED3	ED4	ED5	ED6	ED7	ED8	ED9	EDA	EDB	EDC	EDD	EDE	EDF
EE0	EE1	EE2	EE3	EE4	EE5	EE6	EE7	EE8	EE9	EEA	EEB	EEC	EED	EEE	EEF
EF0	EF1	EF2	EF3	EF4	EF5	EF6	EF7	EF8	EF9	EFA	EFB	EFC	EFD	EFE	EFF

Tags
 Variation Selectors
 Unallocated code points

As of Unicode 15.1

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

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
 - $\frac{1}{4}$ (U+00BC) \rightarrow 1 / 4 (U+0031 U+2044 U+0034)
 - x^3 (U+0078 U+00B3) \rightarrow x3 (U+0078 U+)
 - \mathfrak{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	xxxxyyyyyyzzzzzzwwwwww	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
xxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxx	U+0000 - U+FFFF
110110xxxxxxxxxx 110111yyyyyyyyyy	xxxxxxxxxyyyyyyyyyy + 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, months, weekdays, day periods, time zones
 - Currencies and other units

End of Internationalization

Thank you for listening! 😊