Exchange 2004

From 26 to 96,000 Characters in 60 Minutes

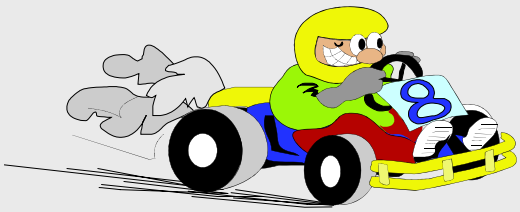
Tex Texin
XenCraft

Objectives

- Modern software integrates with many technologies
- Unicode is the basis for text processing in many environments
- Objective is to inform on many aspects of working with Unicode

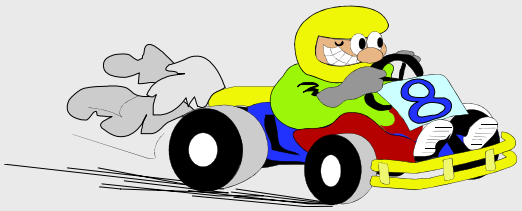
On Your Mark! Get Set! GO!
26 Letters

- ABCDEFGHIJKLMNOPQRSTUVWXYZ



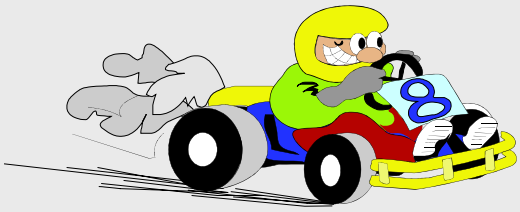
52 Letters, so I don't have to SHOUT!

- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- abcdefghijklmnopqrstuvwxyz



Characters Not Just Letters

- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- abcdefghijklmnopqrstuvwxyz
- Punctuation : . , ; ? ! ( ) - …



Arithmetic

- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- abcdefghijklmnopqrstuvwxyz
- Punctuation : . , ; ? ! ( ) - …
- 0 1 2 3 4 5 6 7 8 9 + − ± * × / ÷ < = > % ‰
- # ¼ ½ ¾ ⅓ ⅔ ⅛ ⅜ Ⅳ iii viii Ⅸ ix Ⅻ

Currency and Business Symbols

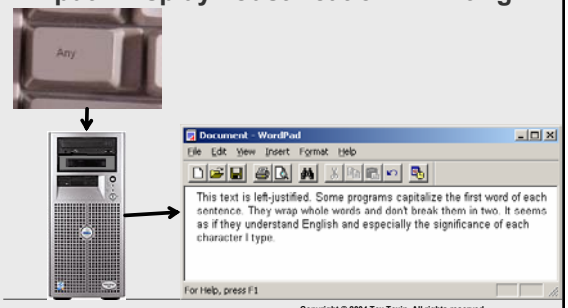
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- abcdefghijklmnopqrstuvwxyz
- Punctuation : . , ; ? ! ( ) - …
- 0 1 2 3 4 5 6 7 8 9 + − ± * × / ÷ < = > % ‰
- # ¼ ½ ¾ ⅓ ⅔ ⅛ ⅜ Ⅳ iii viii Ⅸ ix Ⅻ
- $ ¢ £ ¥ € ₣ £ Pts ₫ ₪ ₩ © ® ™ ℅ ° ℃ ₸ ∞

Common or Specialty Symbols

- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- abcdefghijklmnopqrstuvwxyz
- Punctuation : . , ; ? ! ( ) - …
- 0 1 2 3 4 5 6 7 8 9 + − ± * × / ÷ < = > % ‰
- # ¼ ½ ¾ ⅓ ⅔ ⅛ ⅜ Ⅳ iii viii Ⅸ ix Ⅻ
- $ ¢ £ ¥ € ₣ £ Pts ₫ ₪ ₩ © ® ™ ℅ ° ℃ ₸ ∞
- ¶ § ♀ ♂ ♠ ♣ ♥ ♦ ← ↑ ↓ ↔ ↲ ♩ ♪ ♫ ♫ ♭ ♯ Å
- ∀ ∂ ∃ ∇ ∈ ∉ ∋ ∏ ∑ ≦ ≥ ≡ ≠ ≈ ⊆ ⊇ ☞ ☜

Writing and Computing
How do computers process text?

- Input? Display? Justification? Printing?



Text is a sequence of numbers

- T e x t i s a s e q
- 54 65 78 74 20 69 73 20 61 20 73 65 71
- u e n c e o f n u
- 75 65 6E 63 65 20 6F 66 20 6E 75
- m b e r s .
- 6D 62 65 72 73 2E

"Properties" define behavior for each

- 54 Use the Glyph (image) "T"
- is-character TRUE
- is-digit FALSE
- is-punctuation FALSE
- is-upper TRUE
- tolower 74
- toupper 54
- sort-rank 94

Why is T the number 54 (hex)?

- Arbitrary
- History (ASCII, ISO 646, ECMA 6)
- It isn't always (EBCDIC)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SS	ESC	FS	GS	RS	US
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	'	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	~	]	^	_
60	@	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	¬	}	|	DEL

IBM Extended Binary Coded Decimal Interchange Code (EBCDIC)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Nul		DS		SP											0
1			SOS				/		a	j			A	J		1
2			FS						b	k	s		B	K	S	2
3		TM							c	l	t		C	L	T	3
4	PF	RES	BYP	PN					d	m	u		D	M	U	4
5	HT	NL	LF	RS					e	n	v		E	N	V	5
6	LC	BS	EOB	UC					f	o	w		F	O	W	6
7	DEL	IL	PRE	EOT					g	p	x		G	P	X	7
8									h	q	y		H	Q	Y	8
9									i	r	z		I	R	Z	9
A			SM		¢	!		:								
B					"	$	,	#								
C					<	.	%	@								
D					(	)	-	'								
E					+	;	>	=								
F					⌐	?	"									

Assigning Characters Numeric Values

- Anyone can do it!
  - Computer, Printer, Phone, Monitor, and other hardware vendors
  - Operating systems, font, and other software vendors
  - Governments
  - Standards Organizations (ISO, NCITS, ECMA, JIS, etc.)

But sometimes it is confusing

- Is ellipses … one character or three?
- How many fractions do I need? ¼ ½ ¾ ⅓
- Is "viii" a digit?
- How many characters is ™? ℅?
- Are these writing symbols or drawing?
- ♀ ♂ ♠ ♣ ♥ ♦ ← ↑ ↓ ↔ ↵ ♩ ♪ ♫ ♫ ♭ ♯ ☞ ☜
- Are these: "∏ ∑" mathematical symbols or Greek Letters? Does it matter?

Choosing numeric values

- Often it depends on your application(s)
  - If you are writing a book about music or card games, suits and notes may be a natural and necessary part of text flow.
  - Math programs give "∏ ∑" special semantics. They may have one value for math operators and another for letters.
  - Searching and Sorting are affected.

  Search for "." or "/". Return "… ", or "½"?

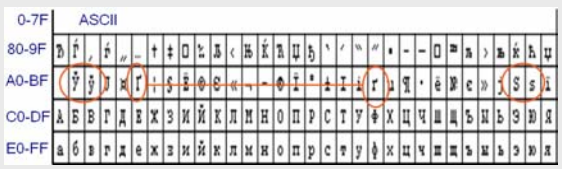
  Search for "8" or "l". Return "viii"?

Choosing numeric values

- Plain text requirements differ from rich text.
  - OpenEdge™
  - OpenEdge<super>TM</super>
- Note- Characters from other languages haven't been discussed yet.

Cyrillic Alphabet

Аа Бб Вв Гг Дд Ее Ёё Жж Зз
Ии Йй Кк Лл Мм Нн Оо Пп Рр
Сс Тт Уу Фф Хх Цц Чч Шш Щщ
Ъъ Ыы Ьь Ээ Юю Яя



Greek Alphabet

- ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
- αβγδεζηθικλμνξοπρςστυφχψω
- Should A, B, E, etc. be same as Latin A, B, E?

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ			
9		‘	’	“	”	•	–	—	˜	™	š	›	œ			Ÿ
A		΅	Ά	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	
B	°	±	²	³	΄	µ	¶	·	Έ	Ή	Ί	»	Ό	½	Ύ	Ώ
C	ΐ	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
D	Π	Ρ		Σ	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	ά	έ	ή	ί
E	ΰ	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
F	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ϊ	ϋ	ό	ύ	ώ	

Small character sets

- Character Set: Collection of “characters” needed in your applications.
- Small sets (<256) require less memory and simpler property implementations.
  - Small number of glyphs
  - Property arrays indexed by numeric value
- May be incomplete and not extensible
  - Language is always evolving
  - ISO 8859-1 is an example that is incomplete

ISO 8859-1 is full, but Euro Needed

- ISO 8859-15 deleted characters
  - acute accent “ ´ ” cedilla “ ¸ ”
  - broken bar “¦” diaeresis “ ¨ ”
  - international currency symbol “¤”
  - Fractions ¼ ½ ¾
- To make room for new ones
  - € Ÿ Š š Œ œ Ž ž
- Similarity to ISO 8859-1 leads to errors

ISO 8859-15

	80	90	A0	B0	C0	D0	E0	F0
			NBSP 00A0	° 00B0	À 00C0	Đ 00D0	à 00E0	ð 00F0
			¡ 00A1	± 00B1	Á 00C1	Ñ 00D1	á 00E1	ñ 00F1
			¢ 00A2	² 00B2	Â 00C2	Ò 00D2	â 00E2	ò 00F2
			£ 00A3	³ 00B3	Ã 00C3	Ó 00D3	ã 00E3	ó 00F3
			€ 20AC	Ž 017D	Ä 00C4	Ô 00D4	ä 00E4	ô 00F4
			¥ 00A5	µ 00B5	Å 00C5	Õ 00D5	å 00E5	õ 00F5
			Š 0160	¶ 00B6	Æ 00C6	Ö 00D6	æ 00E6	ö 00F6
			§ 00A7	· 00B7	Ç 00C7	× 00D7	ç 00E7	÷ 00F7
			š 0161	ž 017E	È 00C8	Ø 00D8	è 00E8	ø 00F8
			© 00A9	¹ 00B9	É 00C9	Ù 00D9	é 00E9	ù 00F9
			ª 00AA	º 00BA	Ê 00CA	Ú 00DA	ê 00EA	ú 00FA
			« 00AB	» 00BB	Ë 00CB	Û 00DB	ë 00EB	û 00FB
			¬ 00AC	Œ 0152	Ì 00CC	Ü 00DC	ì 00EC	ü 00FC
			- 00AD	œ 0153	Í 00CD	Ý 00DD	í 00ED	ý 00FD
			® 00AE	Ÿ 0178	Î 00CE	Þ 00DE	î 00EE	þ 00FE
			¯ 00AF	¿ 00BF	Ï 00CF	ß 00DF	ï 00EF	ÿ 00FF

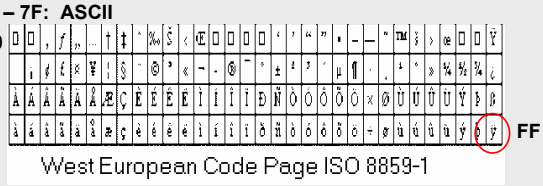
If a character is not in the current code page, how can it be expressed?

If a file contains a 255, what character does it represent?

Characters are not in sorted order

0 – 7F: ASCII

80



FF

West European Code Page ISO 8859-1

FOR EACH CUSTOMER WHERE NAME <= CHR(255):

This does not return all records.

For example, it misses names beginning with "Z" or "z".

Does language determine encoding?

- In Spanish, "ch" is a letter.
- It sorts after "c" and before "d"
  - Color, Charlar, Dar
- "ch" is not (usually) encoded separately.
- These ligatures are:
  - IJ ij Œ œ Æ æ ff fi fl ffi ffl

Bidirectional Scripts

- Hebrew, Arabic, Urdu and other languages are written right to left.
- Often they are mixed with left to right text.
- Encodings can be visual or logical
  - Visual – text stored as displayed
  - Logical- text stored as logically advancing, but may display differently.

Visual vs. Logical

Direction

- Direction is sometimes ambiguous.
  - The number 764 in Hebrew is: תשסד.
  - (764, תשסד)
- Characters have a directional property
  - Strong, Weak, Neutral
  - Neutral chars take direction of the "run".
  - Special override characters exist.
- Some Hebrew characters have 2 forms:

כ ך צ ץ פ ף נ ן מ ם

Arabic has more complex rendering

- Right to left like Hebrew
- Both have diacritic marks
- Characters can have 4 shapes:
  - Isolate , Initial, Medial, Final
  - Beh: ب ـب ـبـ بـ
- Complex rendering
  - lam ل alef ا lam-alef ﻻ
  - kashida (elongation of connectors for justification)

Arabic example page from Quran

Japanese, Chinese, Korean

Languages with >255 characters
- Japanese ~ 50,000 characters
- Korean 2172 Hangul
- Simplified Chinese ~ 6000+
- Traditional Chinese ~ 17000 encoded
- GB 18030 ~ Unicode ~ 70,000 Ideographs

Writing direction
- Vertically, columns going right to left.
- Sometimes columns left to right.
- Or written horizontally left to right.

A b c 日 本 語 d e

Japanese, Chinese, Korean

- Spaces are not used for word separation.
- No upper or lower case.
- Sorting is done a number of ways
  - binary, stroke-radical, radical-stroke, phonetic.
  - dictionaries disagree on stroke counts and orderings
  - 日 + 月 = 明
- Input methods needed for large character set
- Intricate characters require taller fonts
- Trie and other techniques for efficiency with large or sparse data are used.

Japanese Encodings

Several approaches to large character sets
- JIS X208-1990 Character Set
- 6879 character grid 95x95 (33-126x33-126)

Encoded as ISO 2022-jp, SJIS, EUC-JIS
- ISO 2022-jp shift-sequenced (aka escape sequenced) 7-bit
- EUC-JIS 8-bit conversion of JIS x208 values (add 128 to each byte)
- SJIS Shift-JIS complex reassignments plus halfwidth katakana

Escape-sequenced character sets

- Escape sequence changes character set
  - Shifter hex values character set
  - Esc (B ASCII
  - Esc (J JIS Roman (JIS X 0201-1976)
  - Esc $ B JIS X 0208-1983
- ISO 2022 defines sequences for other non-Japanese character sets as well.
  - Esc(B English Text Esc$B Nihongo…

Double-byte Character Sets (DBCS) Japanese, Chinese, Korean

Mixed size characters: 1 or 2 bytes
ASCII compatible

	A	b	c	日		本		語		d	e
Text:	S	S	S	L	T	L	T	L	T	S	S
# Chars:	1	2	3	4	4	5	5	6	6	7	8
# Bytes:	1	2	3	4	5	6	7	8	9	10	11

How long is a DBCS string?

ABC123456789 A=61

ＡＢＣ１２３中區ΦΨ A=A2CF

Double-byte Character Sets

- There are many double-byte character sets
  - Shift-JIS for Japanese,
  - EUCJIS for Japanese
  - GB2312, CP936 for Simplified Chinese,
  - KSC 5601, CP 949, CP 1361 for Korean,
  - Big-5, CP950 for Traditional Chinese

Unicode

- Created and maintained by consortium www.unicode.org
- Unicode covers all major living scripts
- Version 4.0 has 96,000+ characters
- Capacity for 1 million+ characters
- Unicode Character Set = ISO 10646
  - Unicode adds character properties and algorithms
  - ISO and Unicode work together to synchronize
  - ISO support enhances international acceptance

Unicode Characteristics

- 16 bit design originally
- Now has 3 equivalent forms
  - UTF-8: 8-bit variable width, multi-byte (max. 4)
  - UTF-16: 16-bit, variable width, surrogates (max 2)
  - UTF-32: 32-bit, fixed width (max 1)
- Designed to avoid multi-byte performance problems
- Precise algorithm specifications provide interoperability
- Allows one binary program image to be used worldwide
- Developers do not need to be linguists to implement

Unicode is Generative

- Character composition can create "new" characters
- Base character + non-spacing character(s)

A + ° = Å

U+0041 + U+030A = U+00C5

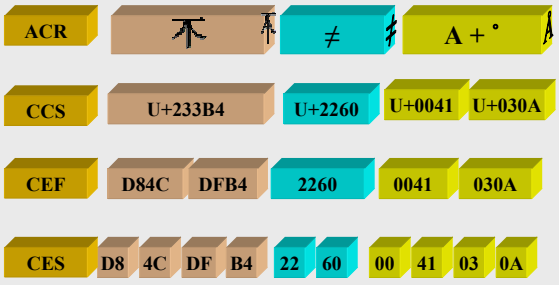
Unicode Characteristics

- Multilingual
  - All scripts, one encoding
- Semantic properties
  Case, digit, alpha/letter/ideogram, directional class, mirroring, combining class, etc. provided by Unicode
- Logical order
  Text is stored in logical order, not visual
- Convertibility
  Accurate round-trip for legacy character sets
- Byte Order Mark (BOM) for endianness, identifier

Unicode — Han Unification

- 70,000 Han characters
  - More than 21 national standards (around 120,000 original characters)
  - Same Chinese root for Kanji, Hanzu and Hanja ideograms
- Drawbacks:
  - No national sort order
  - Disagreement among some scholars

Model for Character Encoding



The Character Encoding Model – Unicode Tech. Report 17

Character Encodings

ACR = Abstract Character Repertoire

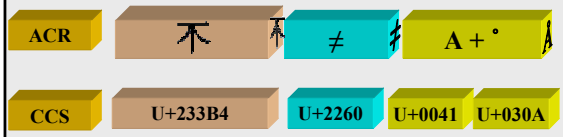
The set of characters you want to be able to represent (aka Character Set).

- Characters can be alphabetic, punctuation, arithmetic, or specific to a discipline or vertical market (e.g. proofreading symbols § ¶, business symbols © ®)
- Characters may be composable.

E.g. Å = A + °

Character Encodings

CCS = Coded Character Set

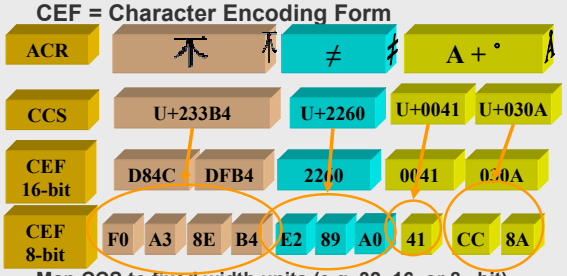


Maps each character to a non-negative unique number.

- Note this example uses hexadecimal numbers.
- The "U+" indicates use of Unicode's numbering.
- The grapheme Å consists of two characters A + °
- Unicode calls these "Unicode Scalar Values"

Character Encodings

CEF = Character Encoding Form

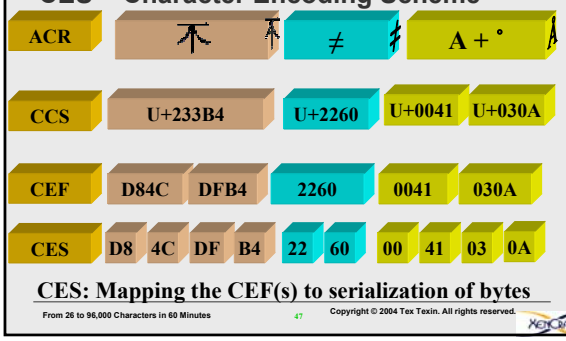


Map CCS to fixed width units (e.g. 32, 16, or 8 –bit)

Note the relationship between the CEFs is not so simple.

Character Encodings

CES = Character Encoding Scheme



CES: Mapping the CEF(s) to serialization of bytes

Character Encodings

- Many character sets exist and in popular use
- Many encoding schemes, even for 1 character set

  ISO 8859-1 ≈ IBM 850

  ISO-2022-JP, Shift_JIS, EUC-JP (JIS X-0208-1997)

  UTF-8 = UTF-16 = UTF-32
- Given just bytes, the character set and the encoding scheme can be indeterminate.
- **Therefore, there are 2 requirements:**
  1) Agreement on names for encodings
  2) Mechanisms for labeling text with encoding

Character Encodings

Names of encodings

IANA (Internet Assigned Numbers Authority) maintains registry of official names for character sets (actually encodings) used on the internet

http://www.iana.org/assignments/character-sets

Character Encodings

- Names consist of ASCII, printable characters
  - case-insensitive, 40 characters max.
  - Aliases are also registered
- Unregistered Names
  - Begin with the name convention "x-"
  - Example: "x-Tex-Yves-encoding"
  - Useful for private encodings or very new encodings
  - Not useful on the web, except for private exchange
- MIME also uses IANA charset names

Character Encodings

- IANA Name Examples (Aliases)
  - ISO_8859-1:1987 (ISO_8859-1, ISO-8859-1, latin1, L1, IBM819, CP819, csISOLatin1)
  - Windows-1252, GB2312, BIG5, BIG5-HKSCS
  - SHIFT_JIS, HP-Legal
  - Extended_UNIX_Code_Packed_Format_for_Japanese
  - Adobe-standard-encoding
  - UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32
- Note- Registry contains many useless names
- Note- Preferred names indicated. Use them.

Storage and Serialization Formats

- UTF-32
  - 32 bits per character
  - Uses Unicode scalar value as-is
  - Big-endian and Little-endian
  - Unicode only goes to 10FFFF (21 bits)
- UTF-16
  - 16 bits per character
  - To support more than 65K characters, uses two "surrogate" values

Properties of UTF-16

Unicode Scalar	UTF-16	
0-FFFF	0-FFFF (Unchanged)	
	High	Low
10000	D800	DC00
10FFFF	DBFF	DFFF

21 Scalar value bits: u uuuu xxxx xxyy yyyy yyyy

wwww = uuuuu – 1

Format: High Surrogate / Low Surrogate

1101 10ww wwxx xxxx / 1101 11yy yyyy yyyy

- Note: Surrogates don't conflict with other values or each other

Properties of UTF-8

- Transforms Unicode to sequences of octets:
  - 7-bit ASCII is in 8-bits with 8th bit = 0
  - All other byte values have 8th bit = 1
  - Non-ASCII characters are either 2, 3 or 4 bytes
- Result:
  - Algorithms searching for ASCII characters (e.g., / \ < > ? + - a b c d etc.) work correctly
  - String length is not greatly increased
  - All of Unicode supported

Properties of UTF-8

- First byte
  - 0XXX XXXX = one byte character
  - 110X XXXX = two byte character
  - 1110 XXXX = three byte character
  - 1111 0XXX = four byte character
- Bytes after first
  - 10XX XXXX

Properties of UTF-8

Unicode Scalar:	x xxxx xxxx xxxx xxxx xxxx			
1-7F:	0xxxxxxx			
80-7FF:	110xxxxx	10xxxxxx		
800-FFFF:	1110xxxx	10xxxxxx	10xxxxxx	
10000-10FFFF:	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Choosing a UTF

- UTF-8
  - Good choice for migrating legacy software and file formats (ASCII compatibility, multi-byte encoding)
  - Best storage form for European languages
- UTF-16
  - More efficient for sorting, processing
  - Best storage for Asian languages
  - Requires wide character datatypes
  - Good choice for new implementations
- UTF-32
  - efficient processing, wastes memory

FAQ

- Does Text Storage Double?
  - Although text expands, not by two
- Performance
  - UTF-8 algorithms are higher performance than DBCS.
  - UTF-16 surrogate characters are rarely used.
  - (But surrogates must be supported.)
- Tip for merging regional databases
  - Remember to realign sequences

UNICODE Byte Order Mark

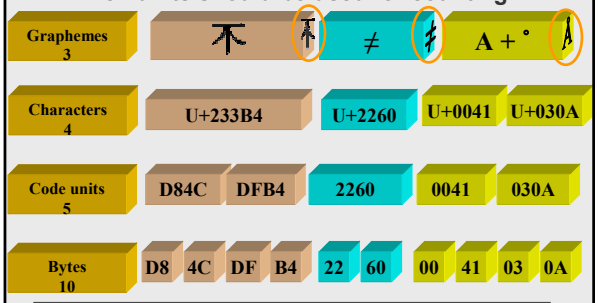
- Indicates byte ordering or endian-ness for UTF-16 and UTF-32
- Identifies data in file as UNICODE
- Specifies UNICODE encoding
  - ASCII 74
  - UTF-16
    - Big Endian **FE FF** 00 74
    - Little Endian **FF FE** 74 00
  - UTF-8 **EF BB BF** 74

When Do You Drop the BOM?

- BOM will be first character in the file
- Be sure to remove before concatenation
- Not needed for UTF-8. But must be accepted.
- Must not be output if encoding is
  UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE

String Indexing Or When 2+2=5?

- Which units should be used for counting?



String Indexing

Character Model indexing recommendations

- Character counting is recommended for most programming interfaces (e.g. XML Path)
- Code unit counting may be used for internal efficiency (e.g. DOM)
- Graphemes may be useful for user interaction, once a suitable definition exists
- Avoid creating API with single unit arguments
  e.g. "SS" = Uppercase("ß")

What is Normal?
Choosing a Canonical Form

- Representing data in more than 1 way leads to errors
- E.g. The Mars Climate Orbiter mission was disastrous. Information expected to be metric, was sent in English units
- Solution- Adopt 1 standard representation- Normalize

Normalization

Unicode characters can have more than 1 representation

- Canonical equivalence
  - Indistinguishable, fundamental equivalence
  - E.g. combining sequences, singletons
  - "Å" U+00C5 (A-ring pre-composed)
  - "A+˚" U+0041 + U+030A (A + combining ring above)
  - "Å" U+212B (Angstrom)
- Compatibility equivalence
  - E.g. Formatting differences, ligatures
  - "ｶ" U+FF76 "カ" U+30AB (KA half and full width)
  - "fi" U+FB01 (ligature fi)

Early Uniform Normalization

- Unicode Consortium has defined canonical and compatibility decomposition formats and 4 different sets of rules for normalization:
  "Unicode Normalization Forms"
  http://www.unicode.org/unicode/reports/tr15/
  - NFD - Canonical Decomposition
  - NFKD - Compatibility Decomposition
  - NFC - Canonical Decomposition, followed by Canonical Composition
  - NFKC - Compatibility Decomposition, followed by Canonical Composition
- Each form gives different results

Normalization

- The W3C Character Model recommends Normalization Form C (NFC) for Web
  - Brings canonical equivalences to composed form
  - Leaves compatibility forms as distinct
  - Most legacy text is composed, and is unchanged
- International Domain Names use NFKC
  - so similar looking characters won't be distinct
  - 'Å' U+00C5 vs. 'Å' U+0212B vs. 'A' + '°' U+0041+U+030A
  - 'u'+'¨' vs. 'ü'
  - 'ｱ' vs. 'ｱ' (fullwidth vs halfwidth)
  - 'œ' vs. 'oe'

Web Normalization

Basic principles

- Without agreement on text representation, binary matching and string indexing fail
- Consequences are significant
  - E.g. Comparison of contracts, security
- Most existing text is composed
- Encrypted strings require normalization first
- Producers have information about the strings they create, simplifying normalization.
- Character escapes taken into account

Unicode Vs. Markup

- 96,000+ characters as of Unicode 4.0
  - Should we use them all?
  - Are there any we shouldn't use?
  - Does Unicode's capabilities, needed for plain text, interfere with markup?
- Markup can do some things better than character codes. Not all Unicode characters are needed.

Unicode Vs. Markup

Potential problem areas

- Redundancies impact searching
  - "Å" A-ring "A+˚ " A+ring "Å" Angstrom
- Formatting characters vs. Markup
  - E.g. Bidi controls, interlinear annotation characters
- Characters with style vs. Markup
  - E.g. Superscript, subscript
- Object Replacement Character vs. Markup
  - Better to use markup to include an image

Unicode Vs. Markup

Solution types

- Restrict characters so they cannot be used
- Replace redundancies (normalization)
- Replace with Markup
  - Extensible
  - presentation can be separate from content

Joint W3C and Unicode recommendations in:

"Unicode in XML and other Markup Languages"

http://www.w3.org/TR/unicode-xml/

http://www.unicode.org/unicode/reports/tr20/

Resource Identifiers: URIs, IRIs

- Currently: URIs encode bytes, not characters
- Most ASCII bytes expressed as ASCII
- Other bytes are %HH
- Non-ASCII characters are ambiguous
  - Character encoding is not taken into consideration
- IRI-Internationalized Resource Identifiers
  - Transcode to UTF-8, then encode as URI
  - Adopters: IE, XLink, XPointer, URN, XML, XML Schema
  - http://www.w3.org/International/O-URL-and-ident.html
  - http://www.w3.org/International/iri-edit/draft-duerst-iri-07.txt

Code page conversions

- Many standards are not standard
  - E.g. Variants of SHIFT-JIS
  - Even ASCII has variants
  - Currency symbols often = 5C (backslash)
- Round trip mappings are sometimes not possible
  - eg. encoding has A-ring and Angstrom, the other has one

SHIFTJIS chars that vary

- 0x5C(YEN SIGN),
- 0x7E(OVERLINE),
- 0x815C(FULLWIDTH EM DASH),
- 0x815F(REVERSE SOLIDUS),
- 0x8160(WAVE DASH),
- 0x8161(DOUBLEVERTICAL LINE),
- 0x817C(MINUS SIGN),
- 0x8191(CENT SIGN),
- 0x8192(POUND SIGN), and
- 0x81CA(NOT SIGN).

Code page conversions

Solutions

- Accurately identify the source encoding variation
- Base conversion on application or semantics
  - e.g. Yen in currency, Backslash in Filename
- Adopt internal standard
- Unicode Normalization
  - Identifies preferred encoding values
- Maintain the original with the conversion
  - To enable round tripping

Unicode does not equal internationalization

- Unicode simplifies development
  - Single source code
  - Enables multilingual processing
  - Properties reduce research for each language
- Unicode does not fix all internationalization
  - E.g. Date, time, number and other formats
  - Linguistic processing can require additional algorithms, data (e.g. word breaking)
  - Continue identify, support cultural requirements
  - Conversion to native encodings for interface to legacy software, systems can impose limitations

Unicode Support

- Employed by all modern technologies
- Standard for web technologies
  - Web consortium's Reference Processing Model
  - Java, HTML, XML, DOM, etc.
  - Supported by browsers IE, Mozilla, Opera
- Support on most platforms and databases
  - Internal code set for Windows NT, 2000, XP
  - Supported by Sun, IBM, HP, Oracle, SQL Server
- Supported by development tools and libraries
  - IBM ICU (International Components for Unicode)

Nice Places To Visit

- Հայաստան Armenia
- ᓄᓇᕗᑦ Nunavut
- 中国 China
- مصر Egypt
- ኢትዮጵያ Ethiopia
- Ελλάς Greece
- 대한민국 S. Korea
- Россия Russia

Conclusions

- Unicode is well-supported, ubiquitous, and often required in integrated environments.
- Unicode simplifies working with different languages
- But the large character set requires some additional considerations.