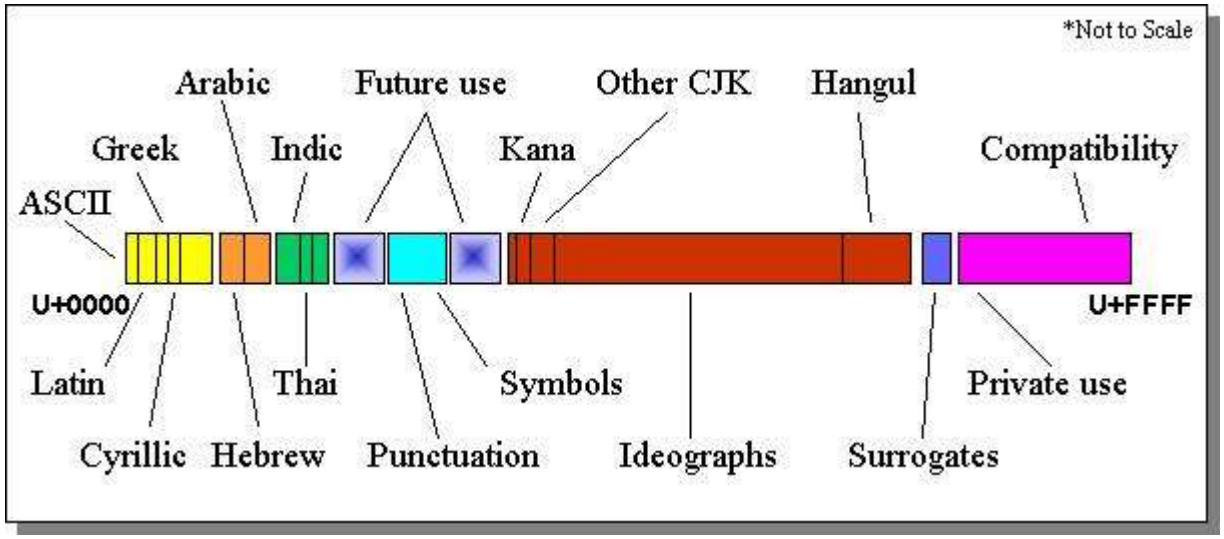


UNICODE

WHAT IS UNICODE AND WHY DO I CARE?

An overview of character encoding

James R. Small, Principal Architect

Michigan!/usr/group

mug.org – A Free and Open Source Michigan Community

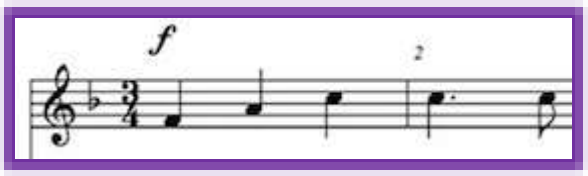
WHAT IS UNICODE?

- Unicode is a standard to represent the world's writing systems
  - » Characters used in written languages

Hello مرحبا 你好 שלום こんにちは

  - » Symbols used in mathematics, music, and other domains (e.g., emojis)

$$S_i(\tau, x, \mu) = G_i(\tau) + \oint\!\!\int \frac{r(x, x', \Theta)}{\varphi(x)} \sum_j P_{ij}(\mathbf{\Omega}, \mathbf{\Omega}') I_j(\tau, x', \mu')\, \mathrm{d}x' \frac{\mathrm{d}\Omega'}{4\pi}$$



  - » Virtually any grapheme (atomic unit of written language/discipline)

WHY DO I CARE?

- Most web sites use Unicode (UTF-8):

UTF-8	89.6%
ISO-8859-1	4.8%
Windows-1251	1.6%
Shift JIS	0.9%
Windows-1252	0.8%
GB2312	0.7%
EUC-KR	0.4%
EUC-JP	0.3%
GBK	0.2%
ISO-8859-2	0.2%
Windows-1250	0.1%
ISO-8859-15	0.1%
Windows-1256	0.1%
Big5	0.1%
ISO-8859-9	0.1%
Windows-1254	0.1%
Windows-874	0.1%

WHY DO I CARE? (WEB UNICODE)

- Just because most web sites use Unicode, why should I care?
  - Do you ever copy and paste to/from a web site?
  - Do you do any web scraping?
  - Do you use any web APIs?
  - Do you interact with any web sites? 😉

WHY DO I CARE?

- The preferred format for E-mail is Unicode (UTF-8)
  - Some E-mail still uses simple ASCII (7 bit text)
  - A large amount of E-mail is HTML-based and thus uses Unicode (UTF-8)
  - The clear direction and standards are towards E-mail with native Unicode support (MIME, RFC 6531/6532)

WHY DO I CARE?


- Windows, MacOS, and UNIX/Linux all use/support Unicode
  - Windows only uses Unicode (UTF-16) for internal character representation and defaults to this for file encoding
  - Unicode is generally not compatible with ASCII*
  - Some tooling only supports simple ASCII

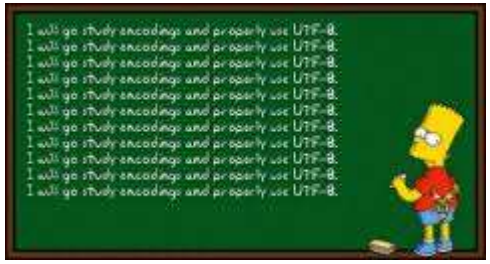
WHY DO I CARE?

- Not understanding Unicode will lead to much suffering:

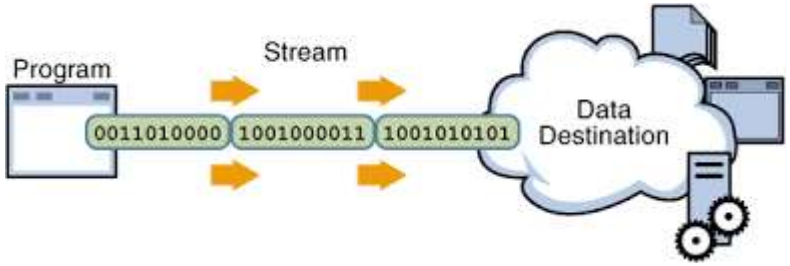
  - »Garbled input/output/displays
  - »Failed deployments/tooling errors
  - »Data loss/corruption
  - »Painful retrofitting

OVERVIEW

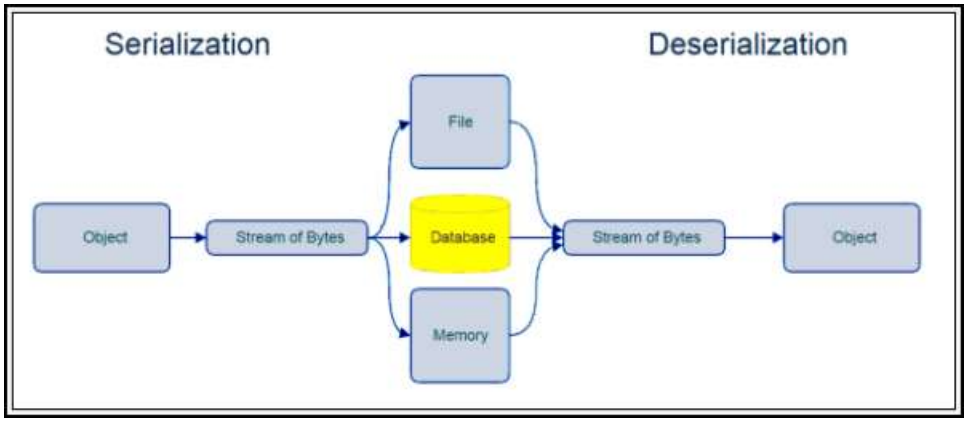
- Prerequisites – A Brief Digression
  - »Serialization
  - »Character Encoding



- Unicode, UCS, and ISO/IEC 10646
  - »A Brief Tour
  - »Unicode Encodings
  - »Examples and Avoiding Problems



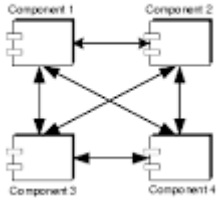
Serialization

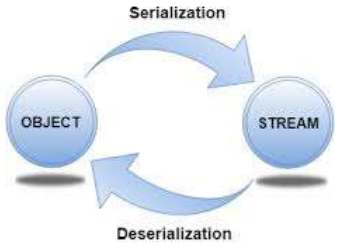


HOW DO APPS COMMUNICATE?

- Using applications/tools

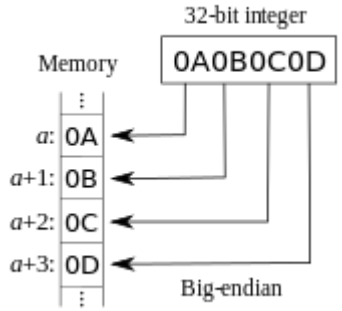
  - »Copy some text from my web browser to my email client
  - »Save a bookmark to a file
  - »Make a query to a database
  - »Instant message someone



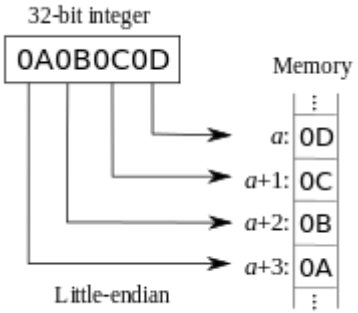


SERIALIZATION COMPLICATIONS

- Byte Order or “Endianness”
  - »Big-Endian – MSB sent first

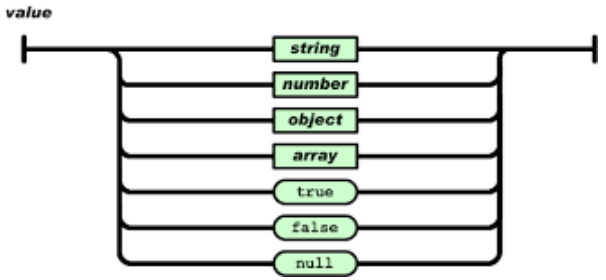


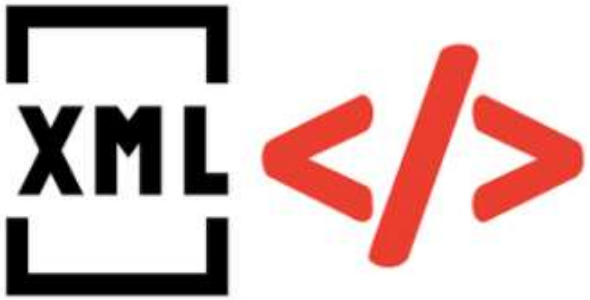
  - »Little-Endian – LSB sent first

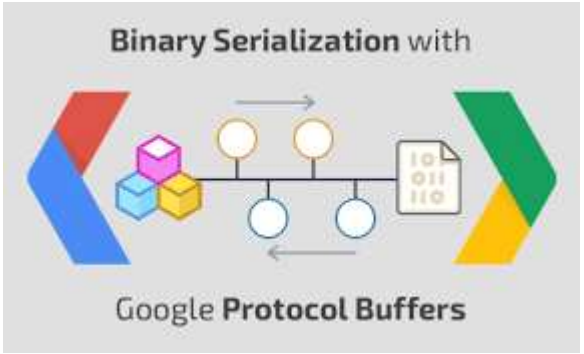


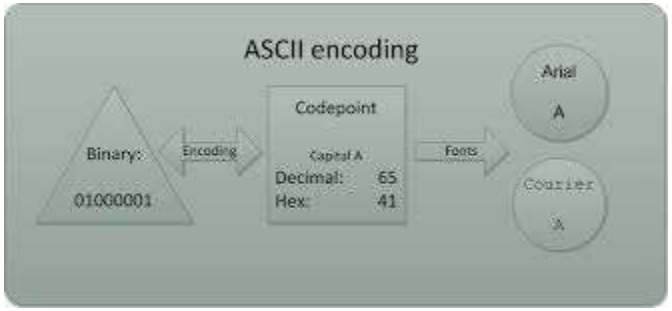
SERIALIZATION FORMATS

- Human Readable
  - » JSON
  - » XML
  - » YAML
- Binary
  - » ASN.1/BER
  - » BSON
- IDLs
  - » Apache Thrift
  - » Google Protocol Buffers

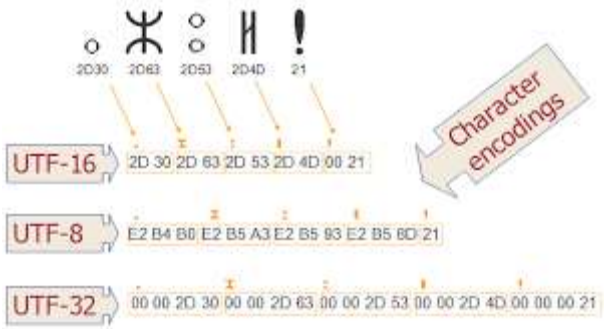








Character Encoding



CHARACTER ENCODING – TERMINOLOGY

Character Encoding	Means to digitally represent discrete characters
Character	Grapheme or atomic unit of a written language/discipline
(Coded) Character Set	Collection of characters used by one or more languages (each assigned unique name & number)
Code Point	The unique number assigned to a particular character within a character set
Code Unit	The bit sequence used to encode a character or code point, e.g., 7 bits for ASCII, 8 bits for ISO-8859-1, 16 bits for UTF-16, 32 bits for UCS-4
Character Map or Code Map	Legacy systems (pre-Unicode) which directly assign a sequence of characters to a sequence of bytes
Character Encoding Scheme	Mapping of code units to a sequence of bytes/octets, e.g., UTF-8, UTF-16, UTF-32/UCS-4

CHARACTER ENCODINGS

- IBM's Binary Coded Decimal (BCD) – 1959
- IBM's Extended BCD Interchange Code (EBCDIC) – 1963
- ASCII – 1963
  - » Only Characters 0-127 are part of the standard! (7-bit encoding)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
							(...)											
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

"EXTENDED" ASCII

- ISO/IEC 8859
  - » Standardized "extended" ASCII
  - » 8-bit encoding
  - » Requires out-of-band method to signal which encoding to use
- Windows-1252 or CP-1252
  - » Like ISO-8859-1 but replaces control characters with display characters

Part	Name
Part 1	*Latin-1, Western European*
Part 2	*Latin-2, Central European*
Part 3	*Latin-3, South European*
Part 4	*Latin-4, North European*
Part 5	*Latin/Cyrillic*
Part 6	*Latin/Arabic*
Part 7	*Latin/Greek*
Part 8	*Latin/Hebrew*
Part 9	*Latin-5, Turkish*
Part 10	*Latin-6, Nordic*
Part 11	*Latin/Thai*
~~Part 12~~	~~*Latin/Devanagari*~~ *(Abandoned)*
Part 13	*Latin-7, Baltic Rim*
Part 14	*Latin-8, Celtic*
Part 15	*Latin-9*
Part 16	*Latin-10, South-Eastern European*

CHARACTER ENCODING – CODE PAGES

ID	Names	Description
37	CP037, IBM037	IBM EBCDIC US-Canada
437	CP437, IBM437	IBM PC US
1250	CP1250, Windows-1250	Latin 2 / Central European
1251	CP1251, Windows-1251	Cyrillic
1252	CP1252, Windows-1252	Latin 1 / Western European
1253	CP1253, Windows-1253	Greek
1254	CP1254, Windows-1254	Turkish
1255	CP1255, Windows-1255	Hebrew
1256	CP1256, Windows-1256	Arabic
1257	CP1257, Windows-1257	Baltic
1258	CP1258, Windows-1258	Vietnamese

MULTIPLE LANGUAGES – CODE PAGES

English: The quick brown fox jumps over the lazy dog.
German: Im finſteren Jagdſchloß am offenen Felsquellwaſſer patzte der affig-flatterhafte kauzig-höfliche Bäcker über ſeinem verſifften kniffligen C-Xylophon.
Polish: Pchnąć w tę łódź jeża lub osiem skrzyń fig.
Greek: ξεσκεπάζω την ψυχοφθόρα βδελυγμία
Russian: Съешь же ещё этих мягких французских булок да выпей чаю.
Spanish: El pingüino Wenceslao hizo kilómetros bajo exhaustiva lluvia y frío, añoraba a su querido cachorro.
French: Les naïfs ægithales hâtifs pondant à Noël où il gèle sont sûrs d'être déçus en voyant leurs drôles d'œufs abîmés.
Hebrew: כיף סתם לשמוע איך תנצח קרפד עץ טוב בגן.
Japanese: いろはにほへど　ちりぬるを
わがよたれぞ　つねならむ
うゐのおくやま　けふこえて
あさきゆめみじ　ゑひもせず

From the Anglo-Saxon *Rune Poem* (Rune version):

ᚠᛇᚻ᛫ᛒᛦᚦ᛫ᚠᚱᚪᚠᚢᚱ᛫ᚠᛁᚱᚪ᛫ᚷᛖᚻᚹᛦᛚᚳᛖᛗ
ᛋᚳᛖᚪᛏ᛫ᚦᛖᚪᚻ᛫ᛗᚪᚾᚾᚪ᛫ᚷᛖᚻᚹᛦᛚᚳ᛫ᛗᛁᛚᚢᚾᛏ᛫ᚻᛦᛏ᛫ᛞᚫᛚᛁᚪ᛫
ᚷᛁᚠ᛫ᚻᛖᛗ᛫ᚹᛁᚾᛖ᛫ᚠᚱᚩᚱ᛫ᛞᚱᛁᚻᛏᚾᛖᛗ᛫ᛞᚫᚱᛖ᛫ᛗᚪᚳᚢᚱᛋ᛬

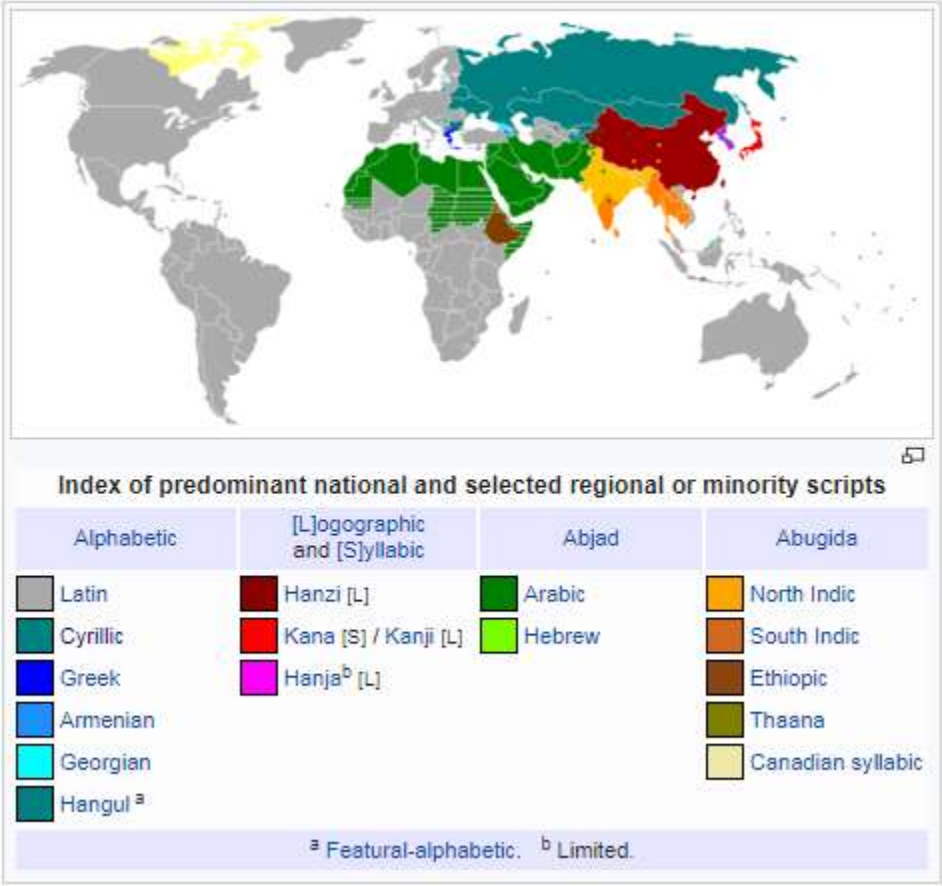
From Laȝamon's *Brut* (*The Chronicles of England*,

An preost wes on leoden, Laȝamon was ihoten
He wes Leovenaðes sone -- liðe him be Drihten.
He wonede at Ernleȝe at æðelen are chirechen,
Uppen Sevarne staþe, sel þar him þuhte,
Onfest Radestone, þer he bock radde.

Kannada poetry by Kuvempu — ಬಾ ಇಲ್ಲಿ ಸಂಭವಿಸು

ಬಾ ಇಲ್ಲಿ ಸಂಭವಿಸು ಇಂದೆನ್ನ ಹೃದಯದಲಿ
ನಿತ್ಯವೂ ಅವತರಿಪ ಸತ್ಯಾವತಾರ

A Brief Tour – Unicode, UCS, and ISO/IEC 10646



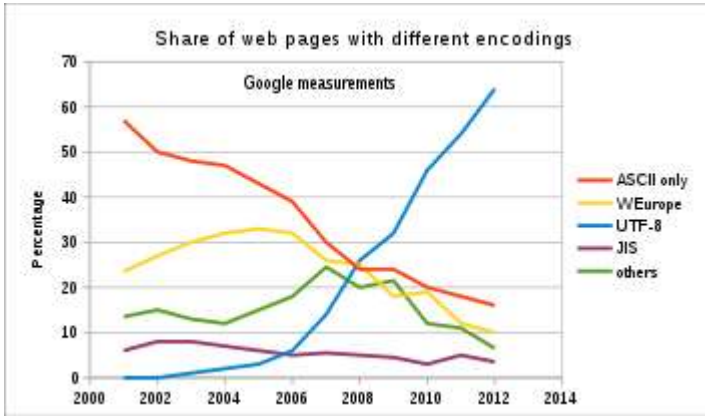
UNICODE ORIGINS

- Originally created for the world's "living" languages
  - » 2^{16} bits or ~65 thousand code points
- ISO created competing standard – 10646, to support all languages
  - » 2^{31} bits or ~2.1 billion code points

- Unified standard (compromise)
  - » 2^{21} bits or ~1.1 million code points
  - » 75% unused – plenty of room

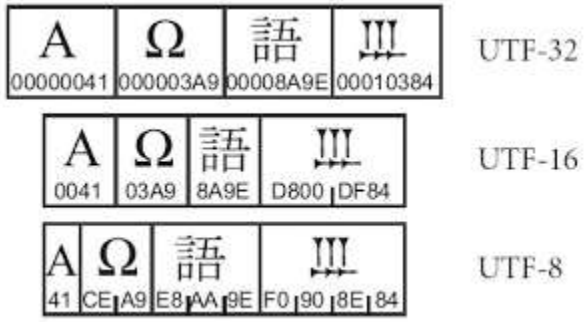
UNICODE/UCS – TERMINOLOGY

Unicode	Industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems
UCS (Universal Character Set)	Standard set of characters defined by the ISO/IEC 10646 — The Universal Coded Character Set
ISO 10646	The UCS standard developed by ISO/IEC JTC1/SC2/WG2 working group
UTF	Unicode Transformation Format – Character encoding (serialization) schemes, e.g., UTF-8, ~~UCS-2~~, UTF-16, UTF-32/UCS-4



Unicode Encodings – UTF-8, UCS-2/UTF-16, UCS-4/UTF-32

Character	UTF-16	UTF-8	UCS-2
A	0041	41	0041
c	0063	63	0063
Ö	00F6	C3 B6	00F6
亜	4E9C	E4 BA 9C	4E9C
𝄞	D834 DD1E	F0 9D 84 9E	N/A



UNICODE MECHANICS

- 2^{21} Code Points – PPHHHH
  - » PP represents 17 Unicode Planes
    - 00 – 10 Hex (00, 01, …, 0E, 0F, 10)
    - Each Plane represents 2^{16} characters
      - 17 * 65,536 = 1,114,112 code points
  - » HHHH represents characters within the plane
    - 0000 – FFFF Hex
- Examples
  - » U+0061 – Latin a
    - No plane indicator implies plane 0 (BMP)
  - » U+1D120 – Musical Symbol G Clef Ottava Bassa
    - Plane 1

UNICODE PLANES

Plane	Description	Allocated Code Pts	Assigned Chars	Special Notes
0 – BMP	Basic Multilingual Plane	65,424	55,294	Surrogate Pairs – D800-DFFF
1 – SMP	Supplementary Multilingual Plane	22,240	20,265	Many useful symbols
2 – SPP	Supplementary Ideographic Plane	60,912	60,859	CJK Characters
3 – 13	Unassigned Planes	-	-	
14 – SSP	Supplementary Special-Purpose Plane	368	337	Non-graphical characters
15 – SPUA-A	Supplementary Private Use Area A	65,536	-	Used by fonts/software
16 – SPUA-B	Supplementary Private Use Area B	65,536	-	Used by fonts/software

UNICODE – EXAMPLE CODED UCS

Char	Code Point	Description	UTF-8	UCS-2/ UTF-16	UCS-4/ UTF-32
a	U+0061	Latin a	61	0061	00000061
ä	U+00E4	a-umlaut	C3 A0	00E4	000000E4
σ	U+03C3	Greek sigma	CF 83	03C3	000003C3
א	U+05D0	Hebrew alef	D7 90	05D0 (BE) D005 (LE)[1]	000005D0 D0050000[1]
٣	U+0663	Arabic digit 3	D9 A3	0663	00000663
カ	U+30AB	Katakana ka	E3 82 AB	30AB	000030AB
退	U+9000	Han Ideograph	E9 80 80	9000	00009000
𠀁	U+21BC1	HKSCS Ideograph	F0 A1 AF 81	NA[2]/ D846 DFC1	00021BC1

Examples and Avoiding Problems

GARBLED/INVALID OUTPUT

- Your app is unable to render the Unicode character – typical substitutions:
  - Replacement Character: 
  - Blank Box: 
  - Gray/Black Box:
- Example:
  - Arabic Letter Heh Goal (06C1): ۀü
  - Arabic Letter Teh Marbuta Goal (06C3): ۃâ
  - Arabic Letter Waw with Ring (06C4): ۄä
  - Arabic Letter Kirghiz Oe (06C5): ۅà

FIXABLE GARBLED OUTPUT

- Linux man page through ssh (terminal):

(…)

```
Description              Octal    latin1    ascii

â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€
              continuation hyphen    255       â€            -
              bullet (middle dot)    267       Â·            o
              acute accent           264       Â´            '
              multiplication sign    327       Ã—            x
```

(…)

- Check Linux locale – probably ends in UTF-8 (what it’s sending) and local terminal (expects) has wrong character encoding

FIXABLE GARBLED OUTPUT

- Linux man page through ssh (fixed):

(…)

Description	Octal	latin1	ascii
continuation hyphen	255	-	-
bullet (middle dot)	267	·	o
acute accent	264	´	'
multiplication sign	327	×	x

(…)

- Local Terminal Session Options, Appearance (SecureCRT) – Changed from Default Character Encoding to UTF-8

TOOLING PROBLEMS

- git, ansible, heroku buildpacks and many tools expect ASCII
  - » If you use Unicode they often act strangely or fail
  - » git typically treats Unicode as a binary file
  - » ansible and heroku buildpacks will throw strange errors

- Search for character codes higher than 0x7F (outside simple ASCII range)
  - » e.g., in VIM or using a regular expression:

```
/[^\x00-\x7F]
```

UNICODE ENCODING DIFFERENCES

- UCS-2 is obsolete but still used by some systems
  - » Predecessor to UTF-16
  - » Doesn't understand surrogate pairs
    - e.g., D846 DFC1
    - These are in the D800-DFFF range:
      - D800-DBFF – High Surrogate
      - DC00-DFFF – Low Surrogate
- UTF-16 is 2 – 4 bytes (typically 2)
  - » If you get weird output your app probably doesn't understand Unicode or UTF-16

UNICODE ENCODING DIFFERENCES

- UTF-8 is 1 – 4 bytes
  - First 128 characters – 1 byte
    - In other words, simple ASCII = UTF-8
  - Next 1,920 characters – 2 bytes
  - Rest of BMP/Plane 0 characters – 3 bytes
  - Other Planes – 4 bytes
- If most of your text is ASCII it will mostly look right with some weird formatting (like garbed man page shown previously)

UNICODE ENCODING CHALLENGES

- Opening a file shows gobbledygook
  - Verify the encoding
  - You may have to specify the correct encoding – often no great way to embed in the file/data
  - Make sure you save files with the correct encoding – saving Unicode files as ASCII will corrupt them
  - You may have to open Unicode files in “binary” or some special mode

QUESTIONS

@sockduct

GitHub Repo

Appendix

REFERENCES – 1

- The Unicode Consortium
- W3Techs - Usage of character encodings for websites
- Unicode and E-mail
- Multipurpose Internet Mail Extensions (MIME)
- RFC 6531 - SMTP Extension for Internationalized Email
- RFC 6532 - Internationalized Email Headers
- Unicode Overview
- Serialization Overview
- Comparison of Data Serialization Formats
- IDL Overview
- Endianness Overview
- Character Encoding Overview
- Code Page Overview

REFERENCES – 2

- C0 and C1 Control Codes Overview (C0 = ASCII, C1 = ISO 6429)
- Windows Code Pages Overview
- ASCII Overview
- Unicode (UTF-8) Sampler Page
- Comparison of Unicode Encodings
- UCS/Universal Coded Character Set Overview
- UCS Characters Overview
- Unicode Planes Overview
- Unicode Character Table
- UTF-8 Overview
- UTF-16 Overview
- UTF-32 Overview

REFERENCES – 3

- UTF-8, a transformation format of ISO 10646 (RFC 3629)
- UTF-16, an encoding of ISO 10646 (RFC 2781)
- Unicode Format for Network Interchange (RFC 5198)
- iconv – UNIX/Linux CLI tool/API to convert between character encodings
- chardet - Character encoding auto-detection in Python, open source based on auto-detection code from Mozilla browser
- Unicode v10.0.0 Announcement and Docs
- The Unicode Standard: A Technical Introduction
- Unicode Tutorial from Adobe – 79 annotate slides, from 2006 but still good
- UnicodeMath - A Nearly Plain-Text Encoding of Mathematics, Version 3.1 (versus application specific encoding like LaTeX)
- List of XML and HTML character entity references

REFERENCES – 4

- Mojibake - the garbled text that is the result of text being decoded using an unintended character encoding
- Unicode Specials block
- An Encoding Primer
- 8-bit clean overview (related to SMTP/E-mail transmission of Unicode/MIME)
- Punycode - a way to represent Unicode within the limited character subset of ASCII used for Internet host names
- Operating System Locale (Used by UNIX/Linux to set character encoding)
- Unicode font overview
- Very basic PowerShell script to detect file character encoding
- Binary-to-text encoding (Encoding 8-bit data into 7-bit ASCII streams)

REFERENCES – 5

- Internationalized domain name overview (Internet domain name using non-ASCII characters, e.g., Unicode)
- What Every Programmer Absolutely, Positively Needs To Know About Encodings And Character Sets To Work With Text
- Programming with Unicode (Open Source Book)
- UCS vs UTF-8 as Internal String Encoding
- Marshalling (can be same or similar to serialization depending on context/language)