



Unicode

Raúl García García

Curso 2024/2025

Universidad San Pablo-CEU

Escuela Politécnica Superior

Campus de Montepríncipe

1


La torre de babel
de los ordenadores

Primer ordenador en 1945
para calcular trayectorias
de misiles.
Para representar letras se
hacia con ASCII.
Originalmente con los
caracteres americanos.

La respuesta original
cuando hacian falta
characters de otros
idiomas te jodes.

En los 80 usaron el ultimo
bit vacio para para usar el
resto de bits para otros
simbolos, pero no cabian
todos los alfabetos. Habia
que hacer muchos
estandares para todos los
alfabetos, se van
acabando.

Pero con el chino y
japones y eso ya era
imposible, simbolo por
palabra, joder...



Unicode principles

- » 21-bit character codes
- » Efficiency
- » Characters, not glyphs
- » Well-defined semantics
- » Dynamic composition
- » Plain text
- » Logical ordering
- » Unification
- » Equivalence
- » Convertibility

18-ene.-25

Curso 2024/2025

página 2


Se hizo unicode como
estandar para unificar a
todos. Cada uno en su
sitio para poder
intercambiar informacion
en cualquier idioma

2

ASCII tiene 128 (7 bits) o 256 si usan el 8 bit.

La mayoría de idiomas usan unos 100-200 characters y en china unos 40000 (100 idiomas * 100 caracteres + chino = unos 50000 characters para Unicode)

El chino y japones usan muchos de los mismos símbolos



Character Codes & Efficiency

» Character Codes

- Unicode 4.0 had 57,129 16-bit characters out of a total maximum of 63,470 10^{16}
- A further 45,718 rare or archaic characters are encoded with two consecutive 16-bit code units from reserved ranges (called "surrogates")

» Efficiency

- No special escape or shift characters required
- All representations of Unicode are self-synchronizing and can be randomly accessed
- Formatting characters are kept to a minimum

18-ene.-25

Curso 2024/2025

página 3

En la v4.0 arqueólogos querían que metieran lenguas muertas.

Así que pasaron de 16 bits a 21 bits para acomodarlo.

No usan caracteres de escape, porque significa que tienes que mirar al de al lado para saberlo, así que codificaron el valor de los caracteres de escape [tab].

En verdad sí que hay alguno


Quieren que los caracteres sean accesibles individualmente dentro de un texto, que no haga falta mirar para los lados para el contexto

3

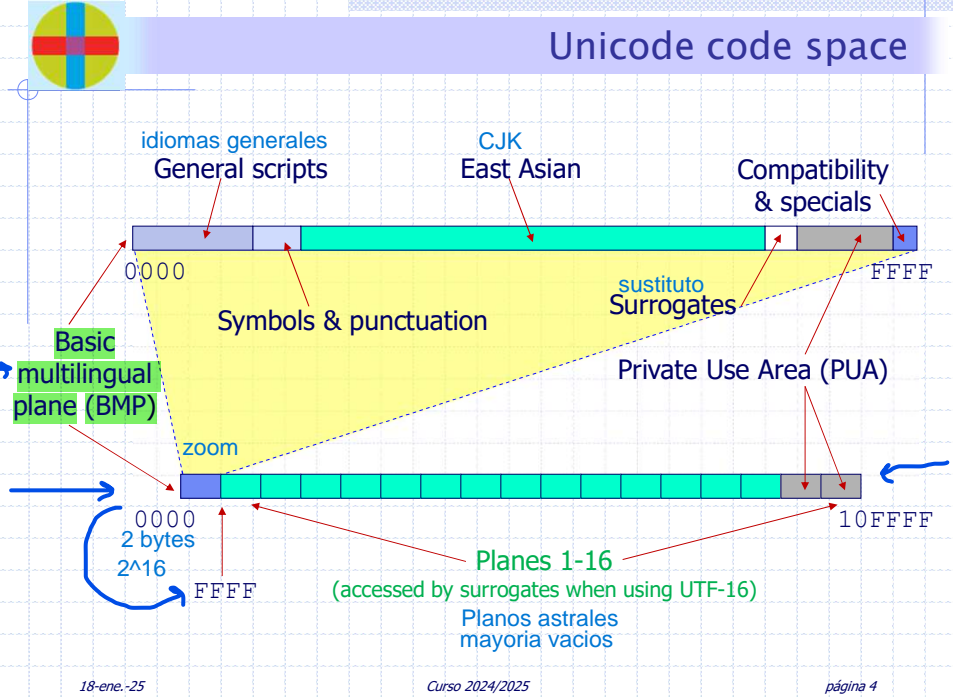
La mayoría de las cajas están vacías, se pasaron bastante

El unicode original
La mayoría de idiomas están aquí

El unicode original era solo la caja azul



Unicode code space



idiomas generales
General scripts

CJK
East Asian

Compatibility
& specials

0000

FFFF

Basic
multilingual
plane (BMP)

Symbols & punctuation

Surrogates

Private Use Area (PUA)

zoom

0000
2 bytes
 2^{16}
FFFF

Planes 1-16
(accessed by surrogates when using UTF-16)
Planos astrales
mayoría vacíos

10FFFF

18-ene.-25

Curso 2024/2025

página 4

En el unicode original habían unos cuantos cientos de PUAs

Para que un programa lo use para lo que le salga de los huecos (pero solo privados)

4


Unicode se encarga de los characters no de los glyphs.

De los Glyphs se encargan las fuentes (pareja de character + glyph)

Un character de unicode no es mas que un numero. Numero 0041 => letra latina 'A' mayuscula.

Antes de 16 bits, ahora de 21 bits.

Almacenar cosas de 21 bits es un lio, porque da a problemas como endianess, ...



Characters vs. Glyphs

- » **Character**: the smallest component of written language that has semantic value.
- » **Glyph**: represents the shape of a character when rendered or displayed.
- » Fonts contain **glyphs**, not characters
- » Latin A and Greek A (alpha) are **distinct characters with the same glyph**
- » Arabic letters need up to **four glyphs** (initial, medial, final, isolated)
- » "f" plus "i" is rendered with a single **merged glyph** in fine typesetting

18-ene.-25

Curso 2024/2025

página 5


Unidades basicas con unidades semanticas
Representacion visual de un character

Los caracteres como tienen semantica, tienen propiedades, devuelven informacion.

Como una 'a' que es letra asi que alfabetico, y 'minusculta' y se sabe su mayuscula.

Asi que se pueden usar en cosas como en lenguajes de programacion

Al character le preguntas cuanto vale si es un numero y da su valor,



Well-defined Semantics

- » Tables generated by the Unicode Consortium give the **properties** of characters
 - Letter, number, punctuation mark, symbol, diacritic, whitespace ...

Ideographic

Alphabetic

Uppercase

Quotation_Mark

不 与

a ξ ۆ َ ِ

A ≡

" ' « » ' ' 『 』

↳3

8↳4

↳5

Numeric_Value

- » Case mapping, Arabic shaping, normalization ...


18-ene.-25

Curso 2024/2025

página 6

Para vars en lenguajes de prog
Pregunta si es letra o underscore, con su valor de lo que es 'numero, letra, simbolo'.

Si le preguntas a un character te puede responder a estas categorias



Unicode General Categories

- » **Letters**: upper, lower, title, modifier, other (syllables, ideographs, etc.)
- » **Numbers**: digit, letter, other
- » **Punctuation**: connector, dash, open, close, initial-quote, final-quote, other
- » **Marks**: non-spacing, enclosing, other
- » **Symbols**: math, currency, modifier, other
- » **Separators**: space, line, paragraph
- » **Other**: control, format, surrogate, private-use

Numeros que son letras como numeros romanos

Hay mas categorias particulares


18-ene.-25

Curso 2024/2025

página 7

7

Lo normalizas y haces un case mapping



Case Mapping & Normalization

- » **Case Mapping**
 - dz ↔ Dz ↔ DZ
 - Heiß → HEISS → heiss
 - ὀσοζ ↔ 'ΟΣΟΣ
 - topkapı istanbul ↔_{tr} TOPKAPI İSTANBUL
- » **Normalization**

ä

= a + ¨

U+00E4

U+0061 + U+0308

 - Equivalent text – equivalent behavior
 - Same display (for supported repertoire)
 - Normalization generates unique forms

le puedes decir a unicode que te plieguen minusculas <=> mayusculas

Se normalizan characters para que se puedan dividir en una separacion entre los acentos y las letras.

Despues puedes tirar los acentos.

Asi se encuentra mucho mas facil.

18-ene.-25


Curso 2024/2025

página 8

8

Solo texto, guarda solo la informacion minima para que se entienda. No guarda cosas como tamanno o cosas de formato (como en rich text)

Del formato se ocupa html



Dynamic Composition & Plain Text

» Dynamic Composition

- There is no character **LATIN CAPITAL LETTER Q WITH CIRCUMFLEX** (Q con gorrito)
 - It can be represented as **LATIN CAPITAL LETTER Q** followed by **U+0302 COMBINING CIRCUMFLEX**
 - COMBINING CIRCUMFLEX** isn't the same character as **ASCII “^”**
- Fonts can have a precomposed glyph for **LATIN CAPITAL LETTER Q WITH CIRCUMFLEX**

» Plain Text

Opuesto Rich Text

- Unicode encodes just enough information for *bare legibility*
- Plain text** is public, standardized, and universally readable
- SGML, HTML, XML** are suitable “**fancy text**” standards to supply structure and formatting to Unicode plain text

18-ene.-25

Curso 2024/2025

página 9

Juntando el codigo de una letra y el gorrito no existe en ningun lenguaje, pero para la matematica puede ser util. Pues se combinan.

Cosas como el HTML son los que le meten mas cosas al texto (es el nivel de presentacion de la red, HTTP es el de sesion)


XML esta mas pensado para que lo use un programa y HTML mas para las personas, pero es muy similar. Como intercambio de info entre maquinas.

El espanol se escribe de derecha a izquierda y de arriba a abajo. Otros lenguajes como el chino son de arriba a abajo y de derecha a izquierda.

Unicode sabe que lenguaje es asi que tiene que almacenar el como se escribe.

Pero si quieres mezclar lenguajes, IBM saldria como MBI.

Si le preguntas a un caracter cual es su orden te lo da (a => left-to-right,...)



Logical Ordering

» With one minor exception, characters are represented in **Unicode in logical order** (the order they are typed or spoken).

- Unicode provides a table-driven algorithm for reordering text into proper reading order, including **mixed directions**

آي.بي.إم. (IBM)، أبل (APPLE)، هيوليت باكارد (Hewlett-Packard)، مايكروسوفت (Microsoft)، أوراكل (Oracle)، صن (Sun) ...

إيزو ١٠٦٤٦ (ISO 10646)

» Text stored in logical order: No special consideration for processing, only for UI and for legacy encoding conversion

» RTL text (mostly Arabic and Hebrew) flows from right to left

» Embedded **numbers** and **LTR text** flow right to left


» Line break preserves reading order

» Selection: Contiguous text **± contiguous display**

18-ene.-25

Curso 2024/2025

página 10



Unification


- » “A difference that *makes* no difference *is* no difference” --Spock of Vulcan
- » If characters look the same, and are from different source standards, they are a **single Unicode character**
 - Common letters, punctuation marks, symbols, and diacritics are unified
 - Differences in language, font, size, and positioning are not represented *Fuentes y esas cosas no representadas*
 - Identical-looking characters (a, alpha) from different scripts are **not** unified
 - Characters that were distinct in a major national or industry standard are kept distinct for round-tripping purposes

Si dos characters son iguales aun de diferentes lenguajes en ASCII y variaciones, en Unicode solo se le hace un character.

No lo mismo a si tienen el mismo glyph, en ese caso si son diferentes

18-ene.-25 Curso 2024/2025 página 11

11



Han Unification

- » Chinese, Japanese, Korean (CJK) all use the 3000-year-old Chinese characters (*hanzi, kanji, hanja*)
 - Each national character set encodes the characters in its own way
- » If it looks similar and is historically the same, Unicode unifies it!
 - Unicode orders Han characters using the traditional Kang Xi dictionary and other dictionaries
- » **Language differences**, which control the choice of fonts, are expressed by a **higher-level protocol**
- » Simplified and traditional characters are *not* unified in Unicode

Muchos characters de los idiomas CJK son iguales. Por las zonas asiáticas sus characters tienen mucha antigüedad.

A lo mejor se escriben algo diferente pero son los mismo characters. Y como antes, si son los mismos, tienen que ir al mismo sitio Unicode.

Las diferencias de como se ven estos lenguajes al final son como fuentes, se pintan por encima (están todos basados en el chino).

Guardan los characters que almacenan los conceptos


Pero hay characters muy tradicionales y específicos (con matices) si que tienen sus propios characters.

python2 usaba unicode1999 y tenía que hacer conversiones que se pegaba unas hostias flipantes.

python1 usaba ASCII

18-ene.-25 Curso 2024/2025 página 12

12



Equivalence & Convertibility

» Equivalence

- Different ways of representing the same characters are equally valid
- Normalization forms allow documents to be compared easily by suppressing irrelevant encoding differences

» Convertibility

- Characters in other character sets can be converted to and from Unicode, usually 1:1
- ASCII and Latin-1 map codepoint for codepoint
- Conversions are done by mapping tables

18-ene.-25

Curso 2024/2025

página 13

Normalizar, como lo de quitar los acentos a palabras para poder buscarlas mas facilmente


Cualquier juego de characters antiguos se pueden convertir a unicode. Tiene tabals de conversion (casi 1a1)

13

Latin15 es el ""moderno""

aqui tienen un cero antes porque son del primer plano, del BMP

U+00xx
U+01xx
...



Unicode Map: Basic Multilingual Plane

» U+0xxx de 0000 a 0FFF => 16^3 => 4092 characters

- ASCII, Latin, Greek, Cyrillic, Armenian, Hebrew, Arabic, Syriac, Thaana, Indic scripts, Thai, Lao, Tibetan

» U+1xxx de 1000 a 1FFF

- Myanmar, Georgian, Hangul, Ethiopic, Cherokee, Canadian Aboriginal, Ogham, Runic, Philippine scripts, Khmer, Mongolian, Limbu, Tai Le, Extended Latin, Extended Greek

» U+2xxx de 2000 a 2FFF

- Symbols
 - › Punctuation, super/subscripts, currency, letter-like, boxes, numerical, arrows, math, technical, OCR, dingbats, Braille
- CJK radicals Cosas basicas de CJK

» U+3xxx de 3000 a 3FFF

- CJK symbols, Hiragana, Katakana, Bopomofo

18-ene.-25


Curso 2024/2025

página 14

En este primer bloque le cabe todo esto! Casi la mayoria

Un monton de lenguajes antiguos y llevas solo gastados unos 8000 characters

14



Unicode Map: Basic Multilingual Plane


- » **U+3400 to U+9FFF** Una barbaridad de characters
 - CJK Unified Ideographs Lo de antes de unificación Han
- » **U+A000 to U+D7A3**
 - Yi, Hangul Syllables
- » **U+D800 to U+DEFF**
 - Surrogates (*no characters*)
- » **U+E000 to U+F8FF**
 - Private Use (PUA) - Private Use Area
- » **U+Fxxx**
 - CJK Compatibility Ideographs
 - Presentation Forms
 - Halfwidth/Fullwidth

Y esto es el BMP
Basic Multilingual Plane

Fuera del BMP a parte de lenguas muertas, ahora se usan estos planos astrales para meter emojis

18-ene.-25 Curso 2024/2025 página 15

15



Unicode Map: "Astral Planes"


- » **U+1xxxx** Primer plano astral
 - Archaic scripts: Linear B, Old Italic, Gothic, Ugaritic, Deseret, Shavian, Osmanya, ...
 - Math alphabets
 - Music symbols (Western and Byzantine)
 - Emojis
- » **U+2xxxx** Cosas raras del CJK
 - Ultra-rare and specialized CJK ideographs
- » **U+30000 to U+DFFFF** Educadamente "están vacíos"
 - Reserved
- » **U+Exxxx**
 - Tag characters
- » **U+Fxxxx and U+10xxxx** Otra zona del PUA
 - Private Use (PUA)

18-ene.-25 Curso 2024/2025 página 16

16

Viene con descripcion para al menos saber que es

Un caracter unicode tiene 21 bits. Almacenar los 21 bits para cada caracter es como se hacia con unicode21.



Unicode properties

como el dni

0041;LATIN CAPITAL LETTER A;Lu;0;L;;;;N;;;;0061;

descripcion

Representative glyph

A

Semantic properties

Code point: 0041

Name: LATIN CAPITAL LETTER A

General category: Uppercase letter (Lu)

Canonical combining class: Standard spacing (0)

Bidirectional category: Left-to-right (L)

Mirrored: no (N)

Lowercase mapping: 0061

Su version minuscula

0061

hay cosas aqui obviadas

esto es lo que tienen metido dentro

Ocupa el size de una letra normal

Los Mirrored son cosas como los parentesis (que tienen cierre) Estos tambien tienen una propiedad de mapeo a su version de cierre

18-ene.-25

Curso 2024/2025


página 17



Encodings

Como guardo 21 bits en un modo de almacenamiento sin saber quien lo ha enviado, y su endianess.

18



Pre-Unicode


- » ASCII is a 7-bit encoding for about 100 characters
- » latin 1 ISO-8859-1 is an 8-bit encoding for about 200 characters directamente en un byte
- » Shift-JIS is a mixed 8/16-bit encoding for about 8,000 characters
- » How to best **encode** Unicode's 2097152 (2^{21}) possible codepoints?

18-ene.-25 Curso 2024/2025 página 19

La mayoría de codificaciones trabajan a nivel de bytes no bits

Si la mayor parte de los caracteres están en el bmp, no debería de interesar el resto. Y hasta 21 se están malgastando bytes.

19



Three Unicode Encodings

- » The Unicode Standard has **Unicode Transformation Formats (UTF)** that are algorithmic mappings from every Unicode code point (except surrogate code points) to a unique byte sequence (i.e. 8, 16 or 32-bits per code point)
 - All encode the same common character repertoire and can be efficiently transformed into one another without loss of data: so they have **equal representation power**
 - All have advantages and disadvantages
- » The three most famous are:
 - **UTF-8**: 8-bit code units — Habrán caracteres que ocupen un byte otros dos y etc
 - **UTF-16**: 16-bit code units
 - **UTF-32**: 32-bit code units
- » All three need at **most 4 bytes** (or 32-bits) of data for each character

18-ene.-25 Curso 2024/2025 página 20

Algoritmos que mapean puntos de unicode a secuencias de bytes

Tienen la misma capacidad de representar todos los caracteres


utf-32 rellena de ceros el resto de bits sobrantes

20

Unidad de 8 bits

Su ventaja es que es inmune a la endianness porque usa bytes. Si se escribe algo en utf-8 da igual quien lo lea, lo leera bien

Es mas lento y mas complicado eso si.



UTF-8

- » Popular for HTML and similar protocols.
- » Way of transforming all Unicode characters into a variable length encoding of bytes (1, 2, 3, or 4 bytes to encode a character)
- » Advantages:
 - The Unicode characters corresponding to the familiar ASCII set have the same byte values as ASCII
 - Unicode characters transformed into UTF-8 can be used with much existing software without modifications
 - No byte-ordering issue
 - Examples:
 - > "A" is 41 (same as ASCII !)
 - > Alpha is CE 91
 - > Katakana "A" is E3 82 A2
 - > Gothic Ahsa is F0 90 8C B0

hay charaters de un byte, dos, tre y cuatro

18-ene.-25


Curso 2024/2025

página 21

Cuando salen raros cosas como la ñ y esas cosas es que el buscador lo esta reconociendo como latin y esta escrito en ascii

21

Siempre un byte mas Menos los ASCII que son siempre un byte



UTF-8 encoding algorithm

128 F=>4 | 7=> 3

- » U+0000...U+007F → aaaaaaa (7 bits) ASCII
 - 1 byte, first high order bit set to 0: B1=0aaaaaaa
- » U+0080...U+07FF → bbbbbbaaaaaa (11 bits) 4+4+3 = 11bits
 - 2 bytes, first 5 bits stored in the first byte and last 6 bits in the second byte: B1=110bbbbbb B2=10aaaaaa
- » U+0800...U+FFFF → ccccbbbbbbaaaaaa (16 bits) el final del BMP →
 - 3 bytes, first 4 bits stored in the first byte, next 6 bits in the second byte, and last 6 bits in the third byte: B1=1110cccc B2=10bbbbbbb B3=10aaaaaa
- » U+10000...U+10FFFF → dddcccccbbbbbbaaaaaa (21 bits)
 - 4 bytes, first 3 bits stored in the first byte, next 6 bits in the second byte, another 6 bits in the third byte, and last 6 bits in the fourth byte: B1=11110ddd B2=10ccccccc B3=10bbbbbbb B4=10aaaaaa

18-ene.-25

Curso 2024/2025

página 22


Pero como voy al carater 7, no puedes ir al byte 7 porque es de anchura variable (a no ser que sea tipo ascii, 1 byte)

Pero los carateres del medio siempre son 10. El prefijo 10 siempre es del medio, te dice que echas para atras. Y el numero de 1s te dice el numero de bytes de ese caracer

Preserva el orden, dos numeros donde uno es mayor que otro, da a una codificacion utf-8 en ese orden. No hay que decodificat utf-8 para ordenar.

Es mas lento que poner los bits a chorro pero es mas facil y en sistemas modernos vale mucho la pena.

22



UTF-8 encoding algorithm

	Binary Format and Split Bytes			
Code Point Range	Byte 1	Byte 2	Byte 3	Byte 4
ASCII U+000000... U+00007F	aaaaaaa			
	0aaaaaaa			
U+000080... U+0007FF	bbbbbaaaaaa			
	110bbbb	10aaaaa		
U+000800... U+00FFFF fin BMP	ccccbbbbbbaaaaaa			
	1110cccc	10bbbbbb	10aaaaa	
U+010000... U+10FFFF	ddccccccbbbbbbaaaaaa			
	11110ddd	10cccc	10bbbbbb	10aaaaa

18-ene.-25


Curso 2024/2025

página 23

23

Cuando era unicode de 16 bits se cogia y se metia directamente los bits.

Pero sufría de endianess



UTF-16

» Popular in many environments that need to balance efficient access to characters with economical use of storage

- It is reasonably compact
- Each BMP character is represented by the obvious 16-bit code unit
- Other characters are represented by two consecutive 16-bit code units using **surrogates**
- Examples:
 - > "A" is 0041
 - > Alpha is 0391
 - > Gothic Ahsa (U+10330) is D800 DB30

18-ene.-25

Curso 2024/2025

página 24


24

Ahora que Unicode tiene 21 bits en vez de 16 como antes, hay mas lio.

tuveron que definir una zona de sustitucion dentro del BMP, los surrogate.

Si te encuentras un surrogate es que no pertenece

Si quitamos el BMP solo nos quedan 20 bits (por eso el surrogate es de 10 bits)



UTF-16 encoding algorithm

» **U+0000...U+D7FF and U+E000...U+FFFF**

- One 16 bit code unit numerically equal to the code point
- The only code points that can be represented in UCS-2


» **U+10000...U+10FFFF**

- Two 16 bit code units called **surrogate pairs**:
 - » 0x010000 is subtracted from the code point, leaving a 20-bit number in the range 0..0x0FFFFF 20 bits
 - » The top ten bits (a number in the range 0..0x03FF) are added to 0xD800 to give the first 16-bit code unit or high surrogate, which will be in the range 0xD800..0xDBFF
 - » The low ten bits (also in the range 0..0x03FF) are added to 0xDC00 to give the second 16-bit code unit or low surrogate, which will be in the range 0xDC00..0xDFFF

» **U+D800...U+DFFF** 800 surrogates (hex)

- The official Unicode standard says that no UTF forms, including UTF-16, can encode these code points

de 10 en 10



18-ene.-25

Curso 2024/2025

página 25

Todos los del BMP se cogen y se meten directamente


Coge el codepoint (del 10000 al 10FFFF)

Por todo este lio, no vale la pena utf-16 casi nunca. Unicos casos son con caracteres JCK porque ocupan la mayoria 16 bts (2 bytes) y en utf-8 son 3 bytes. Aunque ya casi nada

caracter especial que se llama el BOM para los problemas del endianess

Si se escribe mal el BOM, sale FFFE y se lo marca como prohibido y cuando lo detecte recorre y va dando la vuelta a los bytes

No rompe palabra y es de anchura 0 ??



UTF-16 Byte Ordering

» By default, Unicode uses **big-endian**

- This can be overridden by local conventions (e.g. on Windows)

» **UTF-32** Byte Ordering

- **U+0000 U+FEFF**, the **Byte Order Mark** or **BOM**, can be placed at the beginning of a file to unambiguously indicate the byte order, as **U+FFFE U+0000** does not exist

» **UTF-16** Byte Ordering

- Analogously, **U+FEFF**, the **UTF-16 BOM**, can be placed at the beginning of a file to unambiguously indicate the byte order, as **U+FFFE** does not exist

» **UTF-8** BOM

- UTF-8 does not need a BOM to determine byte order
- BOM byte sequence (**EF BB BF**) may still be useful in auto-detecting UTF-8

18-ene.-25

Curso 2024/2025

página 26


00 00 FE FF (vuelta)
FF FE 00 00 (no existe)

Es el utf-16 del siglo 21, poniendo los 21 bits de unicode tal cual, y rellenando el resto a 0s

Tiene tambien el problema de la endianess.

Pero como son de 32 bytes, tiene random access porque para encontrar el character 7, solo lo tienes que multiplicar por 4.

utf-32 tiene sentido en casos donde la endianess no importa, como tu propia maquina. Lenguajes de programacion modernos usan utf-32 como representacion interna para la representacion de caracteres unicode



UTF-32

- » Useful where memory space is no concern, but **fixed width**, single code unit access to characters is desired
 - Each Unicode character is encoded in a **single 32 bit (4 bytes) code unit**
 - Same byte ordering issues as UTF-16
 - Proper subset of **UCS-4** (Universal Character Set 4) in ISO 10646
- » The main advantage of UTF-32, versus variable-length encodings, is that the Unicode code points are **directly indexable**
 - Examining the n'th code point is a **constant time operation**
- » The main disadvantage of UTF-32 is that it is **space inefficient**, using four bytes per code point


18-ene.-25 Curso 2024/2025 página 27

27

Los programas que solo leen ASCII leen bien todo lo que es ASCII del utf-8

Antes era cojonudo porque se metia directamente, pero con los surrogates entre media, ahora obliga a contar

Tiene sentido en tu maquina, en tu memoria



Comparison of encodings


- » Advantages of **UTF-8**
 - Fully ASCII-compatible, including control characters (but not Latin-1 compatible)
 - First byte of any character indicates the number of trailing bytes to follow
 - Sortable, searchable, compressible with 8-bit algorithms
- » Advantages of **UTF-16**
 - *Almost* fixed-width encoding (non-BMP characters are expected to be rare in most documents)
 - As compact as national CJK encodings
 - › UTF-8 costs 50% more
 - Good compromise between space and ease of use
- » Advantages of **UTF-32**
 - Guaranteed fixed-width encoding (directly indexable)
 - Good for internal rather than external (file or network) use

18-ene.-25 Curso 2024/2025 página 28

Gasta un 50% para caracteres chinos y de la zona

Realmente no tiene sentido usarlo ya.

28



Encoding Unicode

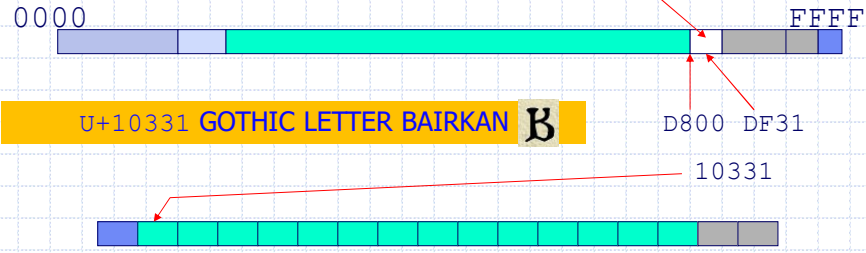
UTF-32 = 10331 (one 32-bit value / code point)

UTF-16 = D800 DF31 (one or two 16-bit values / code point)

UTF-8 = F0 90 8C B1 (one to four 8-bit values / code point)

UTF-16 Surrogates: D800–DFFF


High: D800–DBFF, Low: DC00–DFFF




18-ene.-25

Curso 2024/2025

página 29



¿Preguntas?



18-ene.-25

Curso 2024/2025

página 30