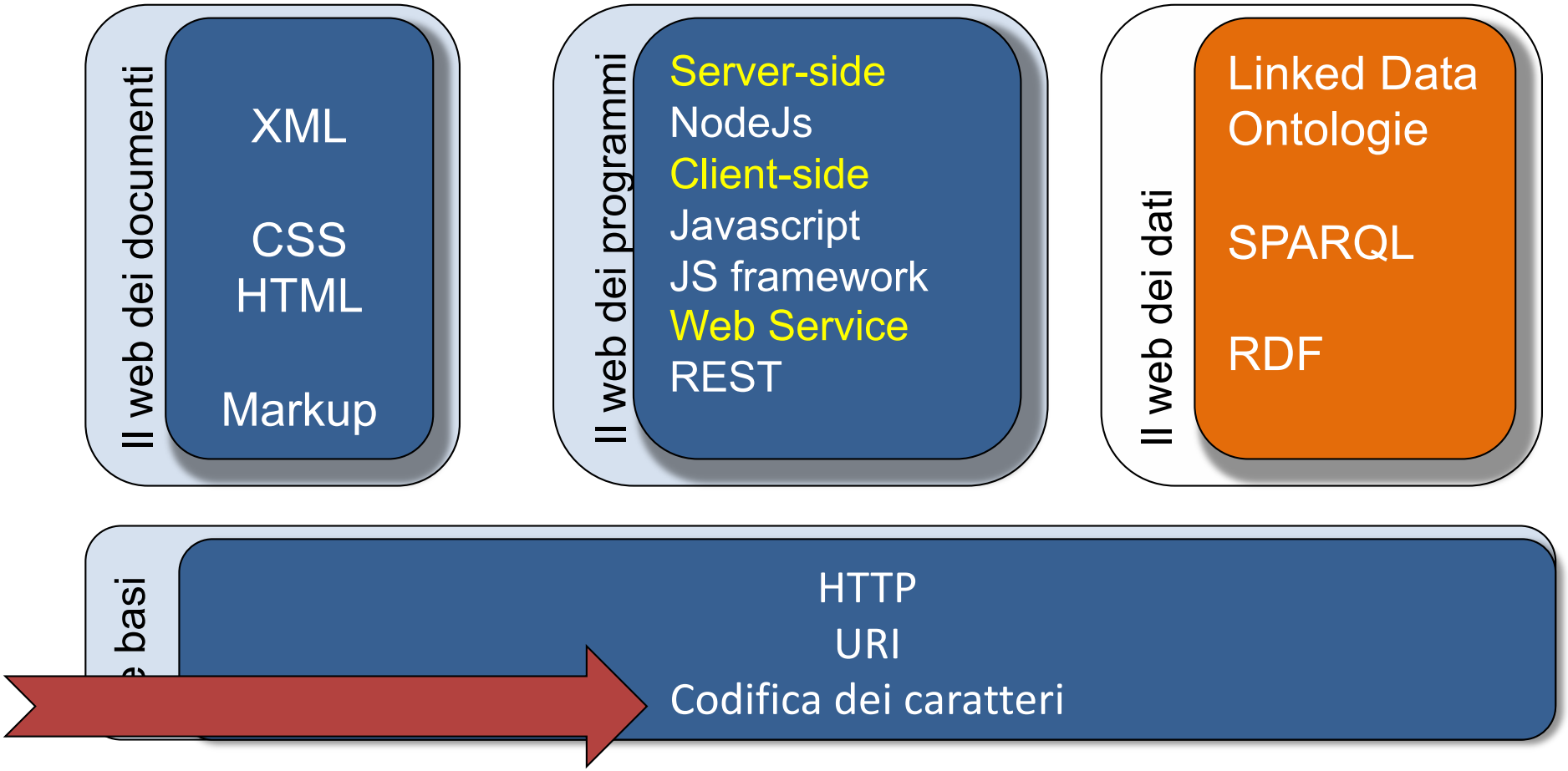

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

# I set di caratteri

**Fabio Vitali**

Corsi di laurea in Informatica e
Informatica per il Management
Alma Mater – Università di Bologna

# Argomenti delle lezioni



# Introduzione

- Qui esaminiamo in breve:
  - Il problema della codifica dei caratteri
  - ASCII (7 bit ed esteso)
  - ISO/IEC 10646 e UNICODE
  - UCS e UTF
  - Content encoding

# La digitalizzazione di dati non-numerici (1)

- Digitalizzare un dato significa associarvi un numero, così da identificarlo e rappresentarne le caratteristiche più importanti.
- Divide et impera: per digitalizzare un dato complesso (ad esempio un testo) possiamo identificarne e digitalizzarne separatamente ogni componente, giù giù fino all'entità atomica ed indivisibile
  - nel caso del testo, l'entità più piccola viene definita come il singolo carattere.
  - Nel caso delle immagini o dei video, la più piccola entità viene trovata nel singolo punto colorato del display (pixel)
  - Più complicato per i suoni, ma non ce ne occupiamo.
- Assegniamo dunque un numero univoco ad ogni carattere diverso del nostro testo e la digitalizzazione dell'oggetto complessivo è dato dalla giustapposizione dei valori associati alle varie lettere.

# I set di caratteri

- La globalizzazione di Internet ha proposto il problema di rendere correttamente gli alfabeti di migliaia di lingue nel mondo.
- Il problema non si pone per i protocolli, che trattano byte interpretati da applicazioni, anche se “per caso” sono significativi per persone di lingua inglese quando scritti in US-ASCII
- Il problema si pone per il contenuto dei protocolli, in quanto deve essere evidente e non ambiguo il criterio di associazione di un blocco di bit ad un carattere di un alfabeto.

# La rappresentazione binaria del testo

- Per rappresentare testo, è necessario fornire una rappresentazione digitale (cioè numerica) degli elementi fondanti della scrittura.
- Identificare gli elementi fondanti (caratteri)
- Identificare lo spazio di rappresentazione
- Creare un mapping e dargli un'ufficialità (standard)

# I caratteri (1)

- Il carattere è l'entità atomica di un testo scritto in una lingua umana.
- In alfabeti diversi i caratteri hanno particolarità diverse:
  - Negli alfabeti di derivazione greca (greco, latino e cirillico), esiste la distinzione tra maiuscole e minuscole, ignota altrove
  - Negli alfabeti di derivazione latina si sono inventati segni particolari sulle lettere per soddisfare le esigenze delle varie lingue che lo usano (accenti, segni diacritici, ecc.).
  - In ebraico, le vocali sono modificatori grafici della forma delle consonanti
  - In arabo, la vicinanza di lettere diverse nella parola provoca un cambiamento nella forma delle lettere stesse.
  - In cinese, è possibile creare nuovi caratteri come composizione di altri caratteri esistenti.
  - … etc…

# I caratteri (2)

Lingue diverse associano ai caratteri ruoli diversi: rappresentano di volta in volta suoni, sillabe, intere parole.

Esistono tre aspetti di un carattere:

- La sua natura (di difficile attribuzione: a e à sono la stessa lettera?)
- La sua forma, o glifo (con ambiguità: P ha un suono negli alfabeti latini, e un altro negli alfabeti greci e cirillici; inoltre i font creano forme anche molto diverse per le stesse lettere).
- Il suo codice numerico: in base ad una tabella piuttosto che un'altra, lettere diverse, di alfabeti diversi, hanno lo stesso codice numerico, o la stessa lettera ha codici diversi.

# Ad esempio


# Lo spazio di rappresentazione

- La digitalizzazione implica l'identificazione di uno insieme di valori numerici da associare ai caratteri.
- Questa associazione può essere arbitraria o secondo regole.
- Le due regole più importanti sono
- ***Ordine***: i valori numerici seguono l'ordine alfabetico culturalmente riconosciuto
- ***Contiguità***: ogni valore numerico compreso tra il più basso e il più alto è associato ad un carattere.
- ***Raggruppamento***: l'appartenenza ad un gruppo logico è facilmente riconoscibile da considerazioni numeriche.
- Es.: in ASCII, i caratteri di controllo hanno la forma 00xx xxxx, i caratteri maiuscoli hanno la forma 10xx xxxx e i caratteri minuscoli hanno la forma 11xx xxxx.

# Altri termini frequenti

- ***Shift***: un codice riservato che cambia mappa da adesso in poi. Lo stesso shift o un secondo carattere di shift, può poi far tornare alla mappa originaria
- ***Codici liberi***: codici non associati a nessun carattere. La loro presenza in un flusso di dati indica probabilmente un errore di trasmissione.
- ***Codici di controllo***: codici associati alla trasmissione e non al messaggio.

# Notazioni binarie ed esadecimali (1)

- La rappresentazione binaria (0b) ed esadecimale (*hex*, 0x) si basano su un sistema numerale posizionale composto da 2 e 16 ($2^4$) caratteri ciascuno.
  - I numeri binari usano solo 0 e 1
  - I numeri esadecimali 0123456789ABCDEF.
- La stessa matematica si applica indipendentemente dal sistema numerale, cambia solo la rappresentazione.

| Decimal | Binary | Hex |
|---|---|---|
| 14 + 25 = 39 | 0b001110 + 0b011001 = 0b100111 | 0x0E + 0x19 = 0x27 |
| 14 = $(1*10^1 + 4*10^0)$ +<br>25 = $(2*10^1 + 5*10^0)$ = | $(0*2^5+0*2^4+1*2^3+1*2^2+1*2^1+0*2^0)$ +<br>$(0*2^5+1*2^4+1*2^3+0*2^2+0*2^1+1*2^0)$ = | 0x0E = $(0*16^1 + 14*16^0)$ +<br>0x19 = $(1*16^1 + 9*16^0)$ = |
| **39 = $(3*10^1 + 9*10^0)$** | **$(1*2^5+0*2^4+0*2^3+1*2^2+1*2^1+1*2^0)$** | **0x27 = $(2*16^1 + 7*16^0)$** |

# Notazioni binarie ed esadecimali (2)

- Poiché $16 = 2^4$, c'è una connessione diretta tra rappresentazione binaria e esadecimale dei numeri secondo cui ogni blocco di 4 cifre binarie (bit) corrisponde ad una cifra esadecimale.
- Quindi un byte composto da 8 cifre binarie è composto da due cifre esadecimali:
  - Per esempio, 0b11001110 può essere diviso in 1100-1110, e poiché 0b1100 = 0xC e 0b1110 = 0xE, ho che 0b11001110 = 0xCE (=206 in decimale).

# Breve ricapitolo di matematica binaria

- 1 bit: $2^1$ combinazioni: 2 valori
- 2 bit: $2^2$ combinazioni: 4 valori
- 4 bit: $2^4$ combinazioni: 16 valori
- 5 bit: $2^5$ combinazioni: 32 valori
- 6 bit: $2^6$ combinazioni: 64 valori
- 7 bit: $2^7$ combinazioni: 128 valori
- 8 bit: $2^8$ combinazioni: 256 valori
- 16 bit: $2^{16}$ combinazioni: 65536 valori o 65k valori
- 32 bit: $2^{32}$ combinazioni: 4.294.967.296 = 4G valori

# Baudot

- Inventato nel 1870 da Emile Baudot, francese, inventore della prima telescrivente (un trasmettitore morse applicato ad una macchina da scrivere)
- Usato nei telex e telescriventi, era un codice a 5 bit, per un totale di 32 codici possibili, ma attraverso l'uso di un codice per lo shift lettere e uno per lo shift numeri, aveva un totale di 64 codici:
  - 50 tra lettere (solo maiuscole), numeri e punteggiatura
  - 9 codici di controllo
  - 2 shift
  - 3 codici liberi
- La codifica non è né contigua né ordinata.

# ASCII

(American Standard Code for Information Interchange)

- standard ANSI (X3.4 - 1968) che definisce valori per 128 caratteri, ovvero 7 bit su 8. Nello standard originale il primo bit non è significativo ed è pensato come bit di parità.
- ASCII possiede 33 caratteri (0-31 e 127) di controllo, tra cui alcune ripetizioni inutili
  - Backspace (sposta la testina indietro di un carattere, utile nelle telescriventi - 08 [0x08]) e Delete (cancella tutti i buchi di un carattere in una scheda perforata, cioè tutti buchi, 1111111 - 127 [0x7F]).
  - Carriage Return (riporta la testina all'inizio di riga - 13 [0x0C]) e Line Feed (gira il carrello di una riga - 10 [0x0A]) che causano molte confusioni nei sistemi moderni.
- Gli altri 95 sono caratteri dell'alfabeto latino, maiuscole e minuscole, numeri e punteggiatura. Codifica contigua ed ordinata. Non ci sono codici liberi.

# EBCDIC

Extended Binary Characters for Digital Interchange Code

- Codifica proprietaria (IBM, 1965) a 8 bit, viene usata nei suoi mainframe. Contemporaneo dell'ASCII
- IBM è molto più sicura della superiorità dei suoi chip, e si azzarda fin dagli anni cinquanta ad usare tutti e 8 i bit del byte.
- 56 codici di controllo e molte locazioni vuote, mentre le lettere dell'alfabeto NON sono contigue, ma organizzate in modo da avere il secondo semibyte che varia da 0 a 9 (0x081-0x89, 0x91-0x099, 0xA1-0xA9, ecc.).

# ISO 646-1991

- Una codifica ISO per permettere l'uso di caratteri nazionali europei in un contesto sostanzialmente ASCII.
- Presenta una International Reference Version (ISO 646 IRV) identica all'ASCII e un certo numero di versioni nazionali
- ISO 646 lascia 12 codici liberi per le versioni nazionali dei vari linguaggi europei. Ogni tabella nazionale la usa per i propri fini.
- I caratteri sacrificati sono: # $ @ \ ¬ ` { | } ~

# Le code page di ASCII

- Quando iniziò ad essere disponibile dell'hardware affidabile vennero proposte delle estensioni di ASCII per usare i rimanenti 128 caratteri (128-255)
- IBM (e, separatamente, Microsoft) proposero un meccanismo di estensioni multiple ed indipendenti di ASCII per le necessità di script, alfabeti ed usi diversi, chiamate code page.
- Esistono svariate centinaia di code page, con qualche carattere in comune a molti diversi.
- Poiché non c'è modo di sapere quale code page è stata usata in un file di testo o in un flusso dati, dobbiamo scoprirlo da fonti esterne.
- Comunque, i caratteri da 0 a 127 sono quelli di ASCII

# CP 737 (PC Code page 737)

- Una code page dei codici per l'alfabeto greco.
- In realtà di code page per il greco ne esistono almeno tre, ma questa era la più popolare.

# KOI7 and KOI8

## Код Обмена Информацией, 8 бит

- Una codifica per i caratteri del cirillico realizzata dal governo sovietico nel 1974
- La versione a 7 bit usa due caratteri switch per passare ai caratteri latini.
- Venne sostituita da Windows 1252 e poi da Unicode

# Arabo
# (PC Code page 720, 864 e 1256)

- Code page per l'alfabeto Arabo.
- Successivamente sostituita da ISO 8859/6

- سلام دنيا

# Codifiche CJK

- Il cinese, il Giapponese e il Koreano (CJK) condividono molti caratteri e per cui tradizionalmente sono stati codificati insieme.
- Poiché i caratteri sono molti di più di 256, non è possibile usare una codifica a 8 bit, ma sono stati usate codifiche a 16 bit (56k combinazioni) e anche più larghe (Unicode al momento codifica 70.000 caratteri di CJK)
- Come minimo, una  ragionevole codifica di CJK deve supportare i caratteri Han (condivisi tra le tre lingue) pù specifici script fonetici come pinyin, bopomofo, hiragana, katakana and hangul
  - Big5 (Taiwan, Macau, Hong Kong), EUC-JP (Japanese), GB18030 (Official People Republic of China encoding), EUC-KR (Korean)

# ISO 8859/1 (ISO Latin 1)

- Estensioni di ASCII sono state fatte per utilizzare il primo bit e accedere a tutti i 256 caratteri. Nessuna di queste è standard tranne ISO Latin 1
- ISO 8859/1 (ISO Latin 1) è l'unica estensione standard e comprende un certo numero di caratteri degli alfabeti europei come accenti, ecc.
- ISO Latin 1 è usato automaticamente da HTTP e alcuni sistemi operativi.
- Ovviamente ISO Latin 1 è compatibile all'indietro con ASCII, di cui è un estensione per i soli caratteri >127.

# L'esigenza di uno standard internazionale

- Esistono dozzine di codifiche a 8 bit per alfabeti non latini (e.g., cirillico, greco e giapponese semplificato) e alcune codifiche a 16 bit per linguaggi orientali (cinese).
- A seconda della codifica usata, posso avere dozzine di interpretazioni diverse per lo stesso codice numerico.
- Debbo ricorrere dunque a meccanismi indipendenti dal flusso per specificare il tipo di codifica usata. Ad esempio:
  - dichiarazioni esterne
  - Intestazioni interne
  - Interpretazione di default delle applicazioni usate
- Ancora più difficile è il caso di flussi misti (un testo italo-arabo, ad esempio), perché è necessario adottare meccanismi di shift da una codifica all'altra, inevitabilmente dipendenti dall'applicazione usata.

# Unicode e ISO/IEC 10646 (1)

- Il compito di creare uno standard unico è stato affrontato indipendentemente da due commissioni di standard, ISO/IEC 10646 (dal 1989) e Unicode (dal 1991).
- Le due commissioni, una industriale, l'altra espressione governativa, hanno lavorato indipendentemente per le prime versioni, salvo poi convergere
- Attualmente la versione 13.0 di Unicode e la versione ISO/IEC 10646:2020 associano esattamente gli stessi codici agli stessi caratteri. Questo però non è garantito nel futuro.
- Sono definite codifiche a lunghezza fissa (UCS-2 e UCS-4), e codifiche a lunghezza variabile (UTF-8, UTF-16 e UTF-32)
- Con la versione del 2020 le differenze residue tra UTF-32 e UCS-4 sono scomparse.

# Unicode e ISO/IEC 10646 (2)

La versione 13.0 (Marzo 2020) definisce 143.859 caratteri diversi, più 140.000 per scopi privati (*Private Use Areas* o PUA), divisi in tre categorie:

- Script moderni
  - Latin; Greek; Cyrillic; Armenian; Hebrew; Arabic; Syriac; Thaana; Devanagari; Bengali; Gurmukhi; Oriya; Tamil; Telegu; Kannada; Malayalam; Sinhala; Thai; Lao; Tibetan; Myanmar; Georgian; Hangul; Ethiopic; Cherokee; Canadian-Aboriginal Syllabics; Ogham; Runic; Khmer; Mongolian; Han (Japanese, Chinese, Korean ideographs); Hiragana; Katakana; Bopomofo and Yi, ecc.
- Script antichi
  - Aegean; Alphabetic and syllabic LTR & RTL; Brahmic; African scripts; Scripts for invented languages; Cuneiform; Undeciphered scripts; North American ideographs and pictograms; Egyptian and Mayan hieroglyphs; Sumerian pictograms; Large Asian scripts;
- Segni speciali
  - punctuation marks, diacritics, mathematical symbols, technical symbols, arrows, dingbats, etc. Discussioni particolarmente infuocate su gli emoji (6.0 – 2010) e sui toni di colore della pelle (8.0 – 2015)

# Star Trek, Lord of the Ring e Unicode

- Star Trek e il Signore degli Anelli fanno parte da lungo tempo della cultura predominante degli informatici occidentali.
- Sia Gene Roddenberry sia J.R.R. Tolkien hanno cercato plausibilità e complessità negli alfabeti da loro inventati. Esistono quindi grammatiche, vocabolari e alfabeti complessi, descritti e noti per Klingon, Ferengi e Elfico (Cirth). È possibile studiarli ed impararli online.
- Nel 1997 venne proposto l'inserimento dell'alfabeto Klingon. Esso venne condizionalmente accettato ed fece parte degli alfabeti in corso di approvazione per quattro anni, e venne definitivamente rifiutata nel 2001, quando venne creata la Private Use Area (PUA) di Unicode:
  - 6400 codici nel Basic Multilingual Plane
  - 65k codici nel piano 15 e 65k nel piano 16.
- Unicode non li assegnerà mai, ma non esiste una assegnazione "ufficiale" di questi codici e chiunque può usarli come crede.
- Tuttavia, almeno per quel che riguarda i codici BMP e del piano 15, c'è un catalogo ufficioso di assegnazioni chiamato ConScript Unicode Registry (scripts for constructed languages), che contiene ad oggi più di 50 script diversi

# I principi di Unicode (1)

- Repertorio universale
  - tutti i caratteri di tutti gli alfabeti
- Efficienza
  - Minimo uso di memoria e massima velocità di parsing. In particolare, raggruppamenti, allineamento e assenza di shift.
- Caratteri, non glifi
  - i font sono completamente esclusi da qualunque considerazione nella specifica del codice (c'è posto solo per un carattere A, indipendentemente dal numero di font esistenti)
- Semantica
  - Ogni carattere possiede un suo significato preciso (la ß tedesca è diversa dalla ß greca) nonché proprietà come direzione, esigenze di spaziatura, capacità di combinazione
- Testo semplice
  - I codepoint rappresentano caratteri di testo semplice, senza descrizioni grafiche o tipografiche (non c'è il bold).

# I principi di Unicode (2)

- Ordine logico
  - le sottosequenze di uno stesso alfabeto seguono l'ordine naturale alfabetico dei parlanti
- Unificazione
  - Caratteri comuni a linguaggi diversi, se possibile, vengono unificati in un singolo codice.
  - Ad es., i caratteri giapponesi e coreani che hanno lo stesso valore in cinese vengono definiti con un'unica codifica.
- Composizione dinamica
  - Alcuni caratteri (in arabo, in cinese, ma anche, banalmente, le lettere accentate o con modificatori degli alfabeti europei) sono composizioni di frammenti indipendenti. Questi frammenti hanno codici indipendenti e vengono creati per composizione.
  - Però in certi casi ricorrere sempre ad un doppio codice per lo un carattere composto è eccessivo. Allora per i più comuni (sanciti da un uso frequente) esiste un codice singolo equivalente, chiamato “sequenza equivalente”

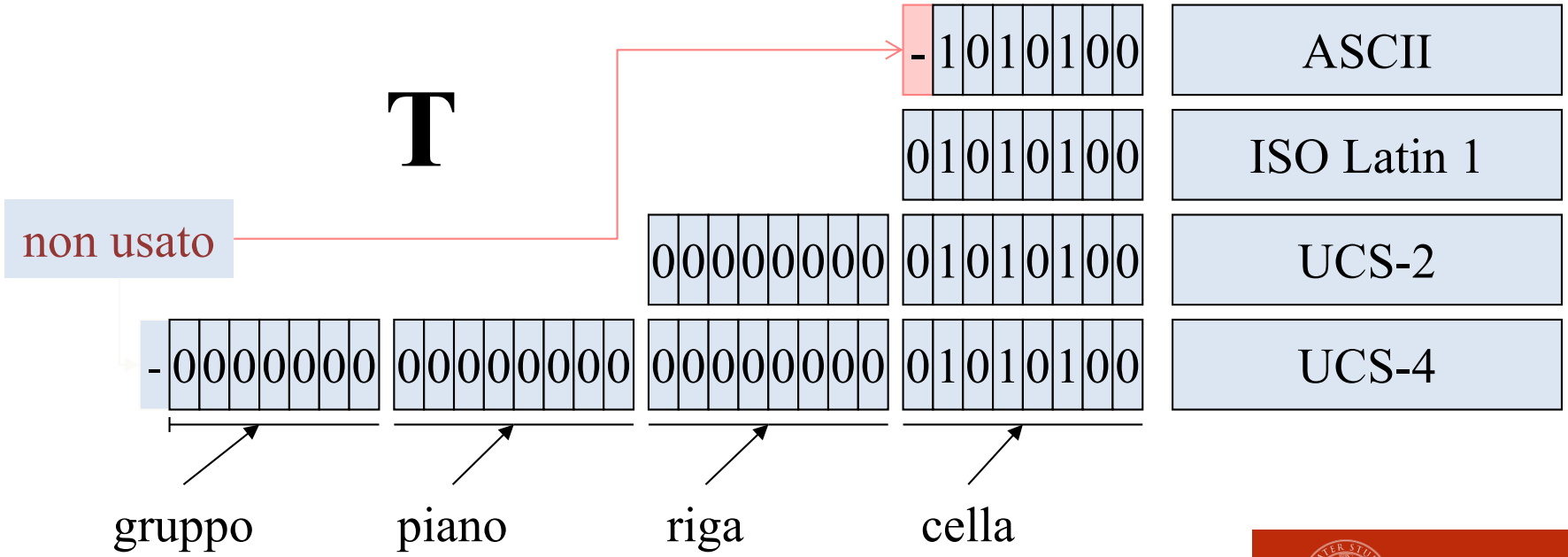
# I principi di Unicode (3)

- Stabilità
  - I codici, una volta assegnati, non possono più essere rimossi e diventano immutabili.
- Convertibilità
  - Esiste un facile meccanismo di conversione tra Unicode e altre codifiche precedenti, in modo da minimizzare gli sforzi di aggiornamento del software.

# ISO/IEC 10646 (1)

ISO 10646 è composto di due schemi di codifica.

- UCS-2 è uno schema a due byte. E' un'estensione di ISO Latin 1.
- UCS-4 è uno schema a 31 bit in 4 byte, estensione di UCS-2. E' diviso in gruppi, piani, righe e celle.



# ISO/IEC 10646 (2)

- In UCS-4 esistono dunque 32768 piani di 65536 caratteri ciascuno. Il primo piano, o piano 0, è noto come BMP (Basic Multilingual Plane) ed è ovviamente equivalente a UCS-2.
- Attualmente sono definiti caratteri sono nei seguenti piani:
  - Piano 0 (BMP o Basic Multilingual Plane): alfabeti moderni
  - Piano 1 (SMP o Supplementary Multilingual Plane): alfabeti antichi
  - Piano 2 (SIP o Supplementary Ideographic Plane): ulteriori caratteri ideografici CJK (Chinese, Japanese, Korean) non presenti in BMP.
  - Piano 3 (TIP o Tertiary Ideographic Plane): caratteri cinesi antichi
  - Piano 14 (SSP o Supplementary Special-purpose Plane): Caratteri tag, in disuso
  - Piani 15 e 16: Private Use Areas

# Da UCS a UTF

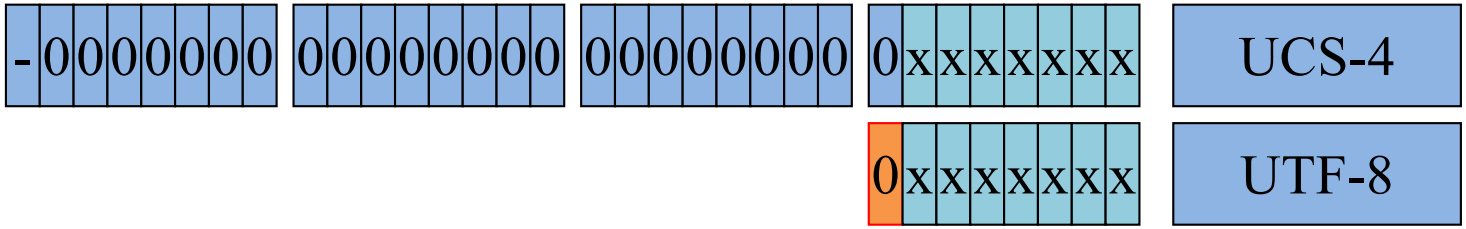
- Nella maggior parte dei casi i testi scritti utilizzeranno soltanto uno degli alfabeti del mondo.
- Inoltre, la maggior parte degli alfabeti sta nel BMP, e la maggior parte dei documenti sono scritti in ASCII.
- E’ dunque uno spreco utilizzare quattro byte per ogni carattere in questo caso.
- In questo caso, sono necessari soltanto una minima parte dei caratteri di UCS.
- UTF (Unicode Transformation Format o UCS Transformation Format) è un sistema a lunghezza variabile che permette di accedere a tutti i caratteri di UCS in maniera semplificata e più efficiente.

# UTF-8 (1)

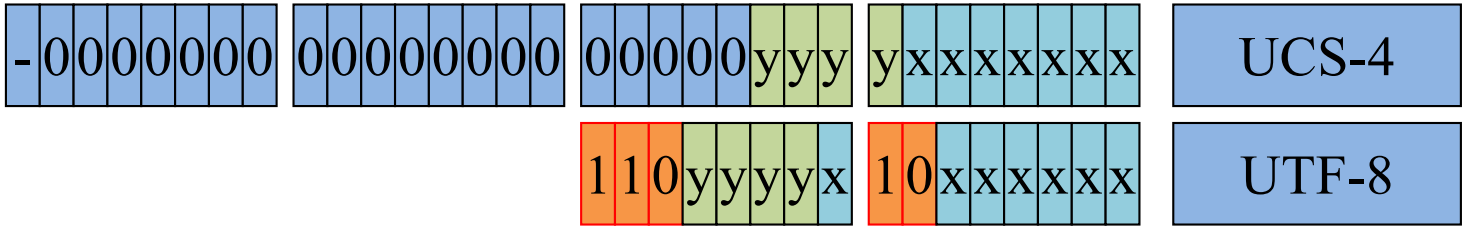
- UTF-8 permette di accedere a tutti i caratteri definiti di UCS-4, ma utilizza un numero compreso tra 1 e 4 byte per farlo.
  - I codici compresi tra 0 - 127 (ASCII a 7 bit), e richiedono un byte, in cui ci sia 0 al primo bit
  - I codici derivati dall'alfabeto latino e tutti gli script non-ideografici richiedono 2 byte.
  - I codici ideografici (orientali) richiedono 3 byte
  - I codici dei piani alti richiedono 4 byte.

# UTF-8 (2)

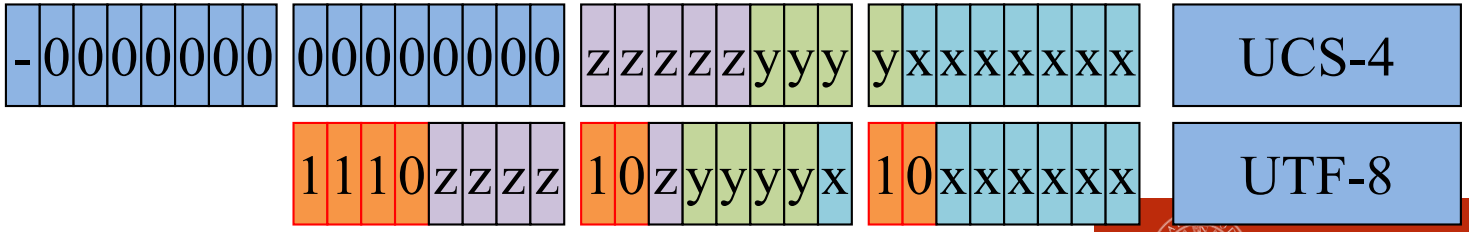
Se il primo bit è 0, si tratta di un carattere ASCII di un byte.




Se i primi due bit sono 11, si tratta di un carattere appartenente ad un alfabeto non ideografico.

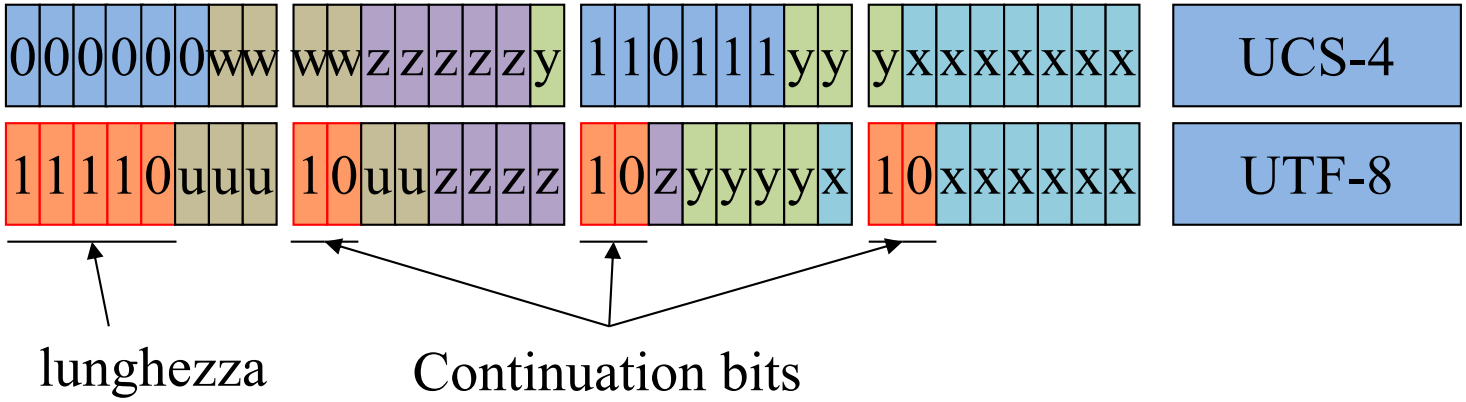



Se i primi tre bit sono 111, si tratta di un carattere appartenente ad un alfabeto ideografico.



# UTF-8 (3)

Se i primi quattro bit sono 1111, si tratta di un carattere che in UTF-16 utilizza coppie di surrogati (caratteri appartenenti ad un piano non BMP ma già precisato).




In generale, il primo byte contiene tanti 1 quanti sono i byte complessivi per il carattere *(lunghezza).*

Il secondo byte e gli altri contengono la sequenza 10 (*continuation bit*) e 6 bit significativi.

Se il byte inizia per 10, allora è un byte di continuazione e debbo ignorarlo fino al primo byte utile.

N.B.: uuuuu = wwww+1 per complesse ragioni

# UTF-8 (4)

Ricapitolando:

| | |
|---|---|
| 0xxxxxxx | È un carattere UTF-8 completo da ASCII |
| 10zyyyyx | È un byte di continuazione |
| 110yyyyx | Primo byte di un carattere di uno script alfabetico |
| 1110zzzz | Primo byte di un carattere di uno script ideografico |
| 11110uuu | Primo byte di un carattere che non sta in BMP ma è già noto |
| 111110uu | Non definito. Apparterrebbe ad un piano non ancora occupato |

# Little-endian, big-endian

- Alcuni processori generano e gestiscono i flussi di coppie di byte ponendo il byte più significativo prima, altri dopo il byte meno significativo.
- Ad esempio, il carattere UTF-16 4F52 sarebbe organizzato come 4F52 su sistemi big-endian (processori Motorola, IBM e in generale RISC), e come 524F su sistemi little-endian (Intel e cloni, DEC, e altri CISC).
- Questo ha degli effetti notevoli sulle capacità di interpretare correttamente flussi di byte provenienti da qualche processore ignoto.
- In particolare, ricevendo un flusso dichiarato UTF-16 o UCS-2, come posso essere sicuro di quale sia il modello di memorizzazione originario?

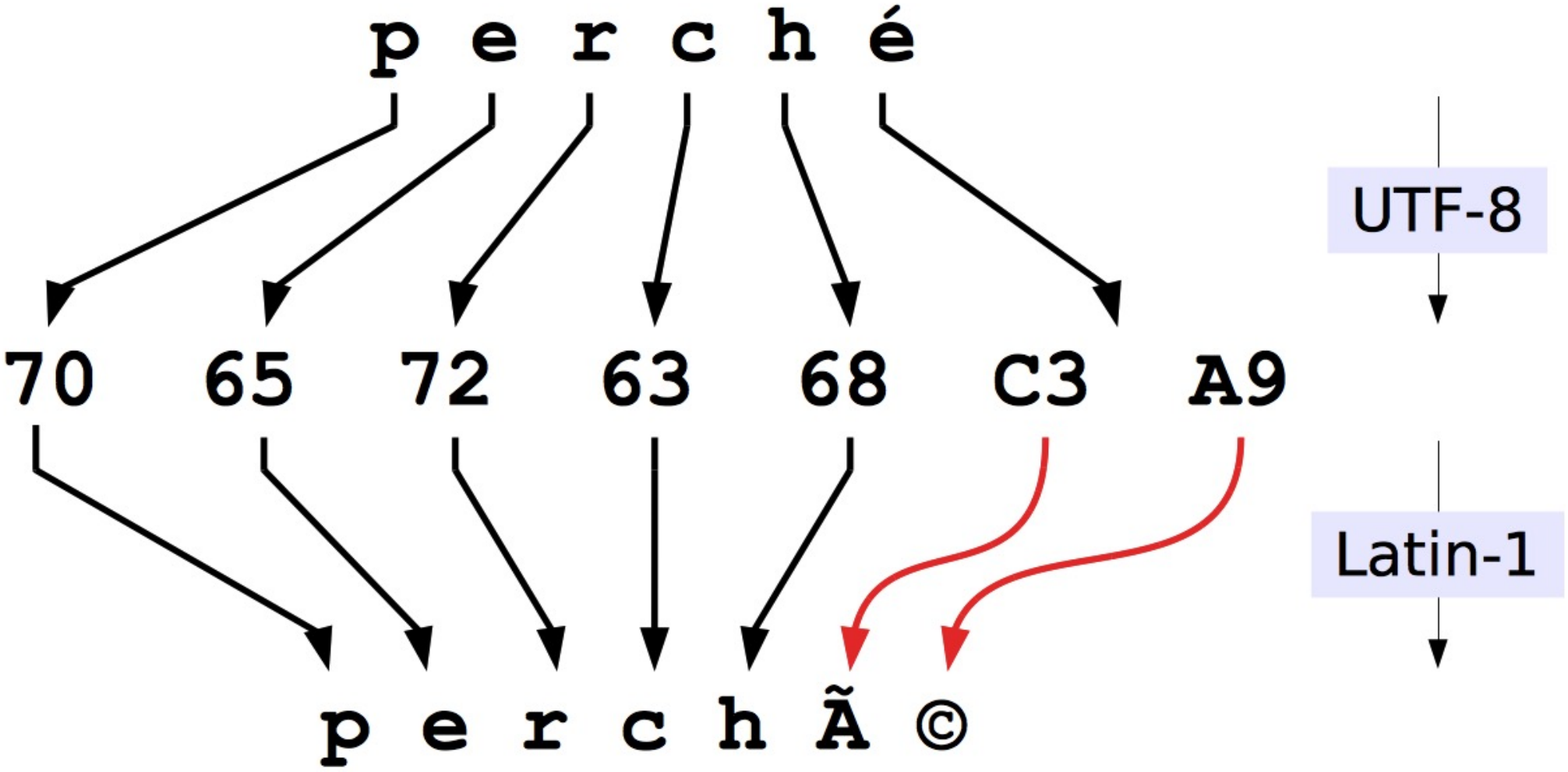
# Byte Order Mark (BOM)

- Unicode specifica un codice, FFFE, come segnalatore di ordinamento del flusso.
- FEFF è il carattere Zero-Width No-Break Space (ZWNBSP), un carattere che può essere usato in qualunque contesto di whitespace (cioè ovunque tranne in mezzo alle parole) senza modificare il significato dei testi. La sua forma corrispondente in little-endian, FFFE, è un carattere proibito in Unicode.
- Unicode suggerisce allora di utilizzare un carattere ZWNBSP all'inizio di ogni flusso UTF-16 e UCS-2. Se il processore riceve FEFF deduce che il sistema sorgente è big-endian, altrimenti che è little-endian, e decide di riconvertire il flusso su questa base.
- Il carattere FEFF usato per questo scopo è allora noto come Byte Order Mark, o BOM. Poiché la conversione da e per UTF-8 deve essere totalmente trasparente, anche molti flussi UTF-8 contengono il BOM.

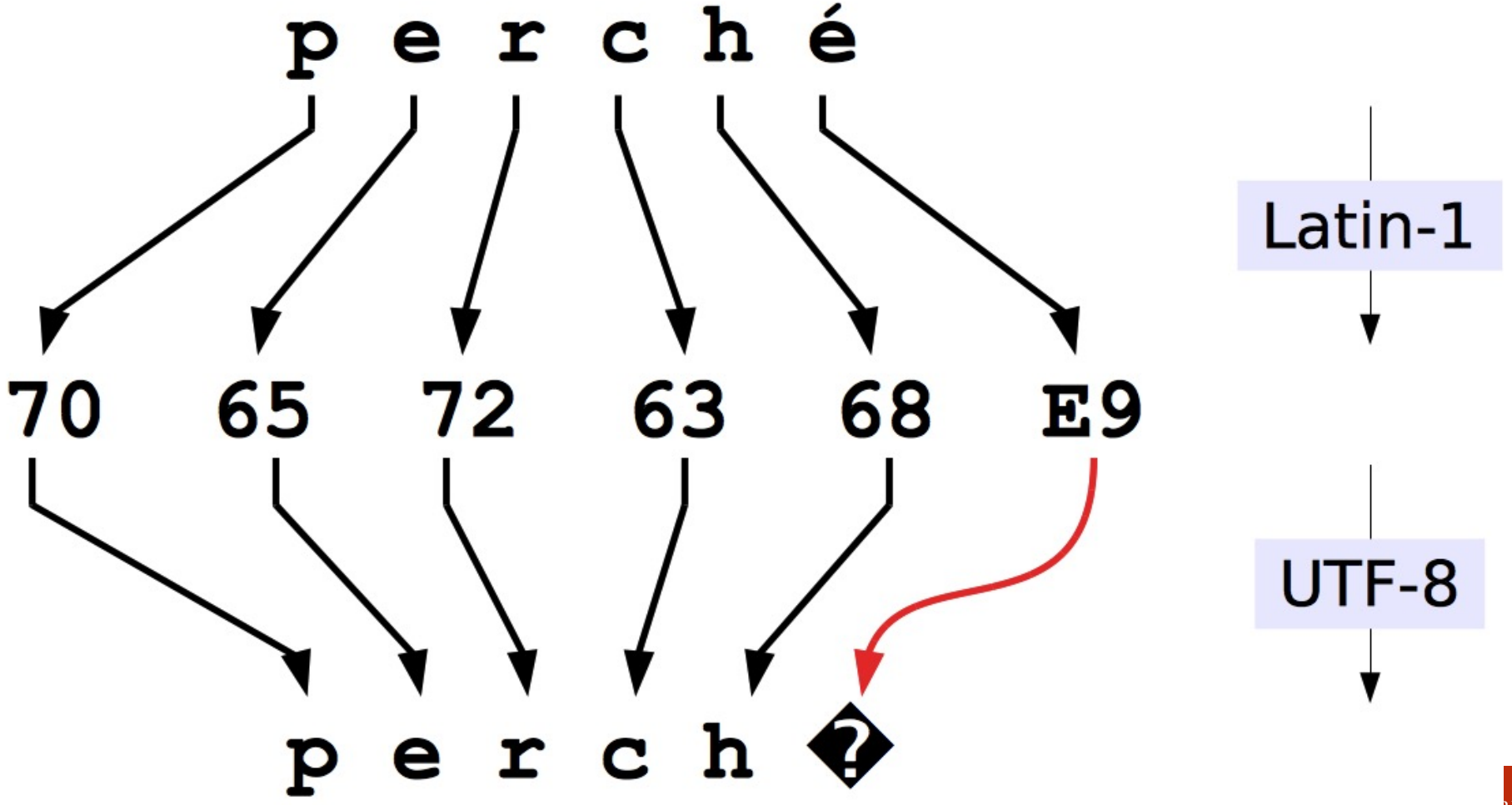
# Differenze tra UTF-8 e ISO Latin-1

- Non confondere le codifiche UTF-8 e ISO Latin-1!
- Per i caratteri appartenenti ad ASCII, le due codifiche sono identiche. Quindi un documento in inglese non avrà differenze nel testo (a parte, ad esempio, virgolette inglesi o trattini, ecc.).
- Viceversa, un testo in italiano, o francese o tedesco risulta quasi corretto, perché non vengono descritte correttamente solo le decorazioni di lettere latine (ad esempio, accenti, umlaut, vocali scandinave ecc.).
- In questo caso, UTF-8 utilizza 2 byte per questi caratteri, mentre ISO Latin 1 ne usa uno solo!

# Problemi comuni (1)



# Problemi comuni (3)



# Un esempio

http://www.filehungry.com/italian

**DBScribe for MySQL - 1.1**
Leadum DBScribe Ã© un strumento grat
DBScribe supporta:
- diversi stili e formati di ...

- -2 la posizione

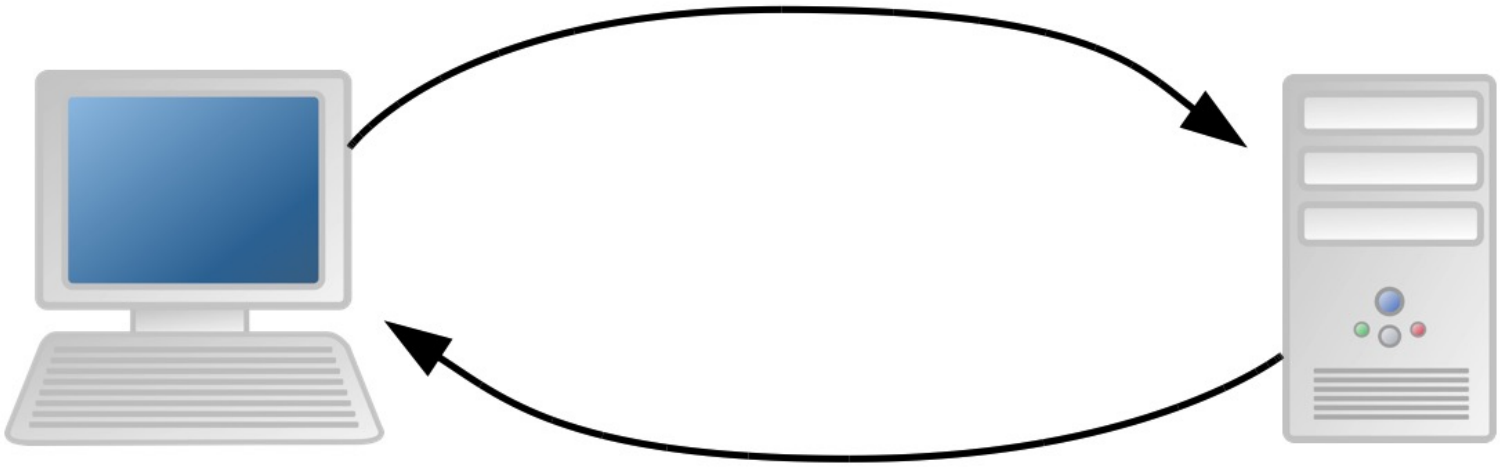
| | | | | | |
|---|---|---|---|---|---|
| 00000yyy yxxxxxxx | UCS-2 | 00000000 11101001 | 00E9 |
| 110yyyyx 10xxxxxx | UTF-8 | 11000011 10101001 | 00E9 |
| | ISO Latin-1 | 11000011 10101001 | C3 A9 |

**Qui sta l'errore:**

Se il carattere UTF-8 00E9 viene interpretato come ISO Latin-1, viene letto come una sequenza di due caratteri, C3 A9, corrispondente a "Ã©"

# Definire la codifica su HTTP

❶
```
GET /user/learco HTTP/1.1
Accept-Content: ISO-8859-1, utf-8
```



❷
```
Content-Length: 1048
Content-Type: text/html; charset=utf-8

<html>…
```

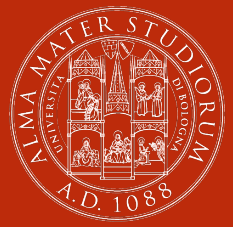
# Conclusioni

Qui abbiamo parlato di set di caratteri

- A lunghezza fissa, 7, 8 bit (ASCII, EBCDIC, ISO Latin 1)
- A lunghezza fissa, 16, 31 bit (UCS-2, UCS-4)
- A lunghezza variabile, 1-4 * 8 bit (UTF-8, UTF-16)

# Riferimenti


- N. Bradley, The XML companion, Addison Wesley, 1998, cap. 13.
- K. Simonsen, Character Mnemonics & Character Sets, RFC 1345, IETF, June 1992
- D. Goldsmith, M. Davis, UTF-7, A Mail-Safe Transformation Format of Unicode, RFC 2152, IETF, May 1997
- The Unicode consortium, Unicode® 8.0.0, Released: 2015 June 17, http://unicode.org/versions/Unicode8.0.0/



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

**Fabio Vitali**

# Corso di tecnologie web

Fabio.vitali@unibo.it

www.unibo.it