# I set di caratteri

Fabio Vitali

# Introduzione

Qui esaminiamo in breve:

- Il problema della codifica dei caratteri
- ASCII (7 bit ed esteso)
- ISO/IEC 10646 e UNICODE
- UCS e UTF



# 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.



# 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 giustapposizione di lettere diverse nella parola provoca una differenziazione della 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.



# 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 Form Feed (gira il carrello di una riga - 14 [0x0D]) 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: # $ @ \ ¬ ` { | } ~



# 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 qualche sistema operativo.
- 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, Unicode e ISO/IEC 10646.
- Le due commissioni, una industriale, l'altra espressione governativa, hanno lavorato indipendentemente per le prime versioni, salvo poi convergere
- Attualmente la versione 3.0 di Unicode e la versione ISO/IEC 10646-1:2000 associano gli stessi codici agli stessi caratteri. Questo però non è garantito nel futuro.
- Per semplicità identifichiamo ISO/IEC 10464 come fonte di codifiche a lunghezza fissa (UCS-2 e UCS-4), e Unicode come fonte di codifiche a lunghezza variabile (UTF-8, UTF-16 e UTF-32)



# Unicode e ISO/IEC 10646 (2)

Al momento definiscono 95,221 caratteri diversi, appartenenti a tre categorie:

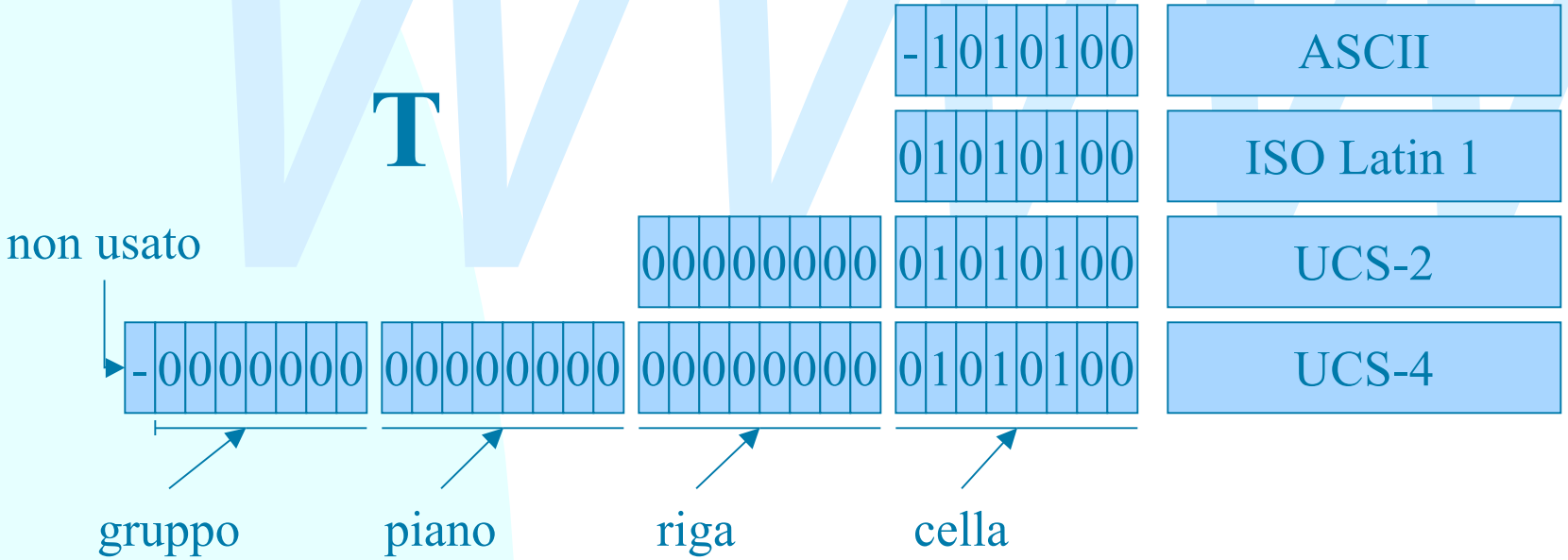
- Script moderni
  - Latin; Greek; Cyrillic; Armenian; Hebrew; Arabic; Syriac; Thaana; Devanagari; Bengali; Gurmukhi; Oriya; Tamil; Telegu; Kannada; Malayalam; Sinhala; Thai; Lao; Tibetan; Myanmar; Georgian; Hangul; Ethiopic; Cherokee; Canadian-Aboriginal Syllabics; Ogham; Runic; Khmer; Mongolian; Han (Japanese, Chinese, Korean ideographs); Hiragana; Katakana; Bopomofo and Yi
- 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



# ISO/IEC 10464 (1)

ISO 10464 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): tutti gli alfabeti moderni
  - Piano 1 (SMP o Supplementary Multilingual Plane): tutti gli alfabeti antichi
  - Piano 2 (SIP o Supplementary Ideographic Plane): ulteriori caratteri ideografici CJK (chinese, japanese, Korean) non presenti in BMP.
  - Piano 14 (SSP o Supplementary Special-purpose Plane): Caratteri tag



# I principi di Unicode 3.0 (1)

Repertorio universale

- tutti i caratteri di tutti gli alfabeti

Ordine logico

- le sottosequenze di uno stesso alfabeto seguono l'ordine naturale alfabetico dei parlanti

Efficienza

- Minimo uso di memoria e massima velocità di parsing. In particolare, raggruppamenti, allineamento e assenza di shift.

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.

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)



# I principi di Unicode 3.0 (2)

## 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.

## Sequenze equivalenti

- 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.

## 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

## Convertibilità

- Esiste un facile meccanismo di conversione tra Unicode e altre codifiche precedenti, in modo da minimizzare gli sforzi di aggiornamento del software.



# 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-16

Il più semplice degli UTF è UTF-16.

Esso permette di accedere a tutti i caratteri di UCS-2 (o BMP) in 16 bit, e a tutti i caratteri degli altri piani con due caratteri da 16 bit,

Si basa sull'uso di surrogati, coppie di valori a 16 bit appartenenti a sezioni non utilizzate in altro modo.

- Surrogato alto: codici compresi tra D800 - DBFF
- Surrogato basso: codici compresi tra DC00 e DFFF



# UTF-8 (1)

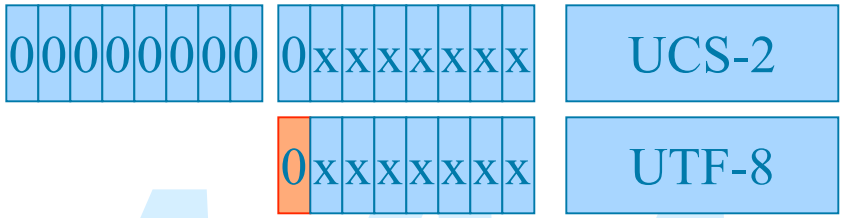
UTF-8 permette di accedere a tutti i caratteri definiti di UCS-4, ma utilizza un numero compreso tra 1 e 4 byte per farlo.

- 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.

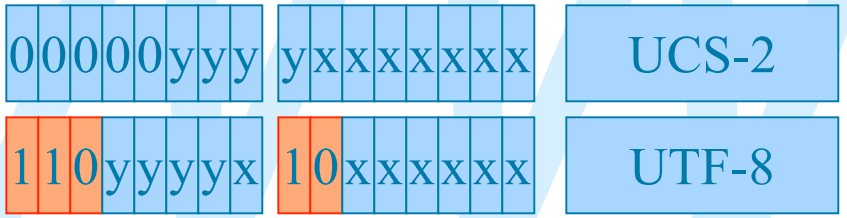


# UTF-8 (2)

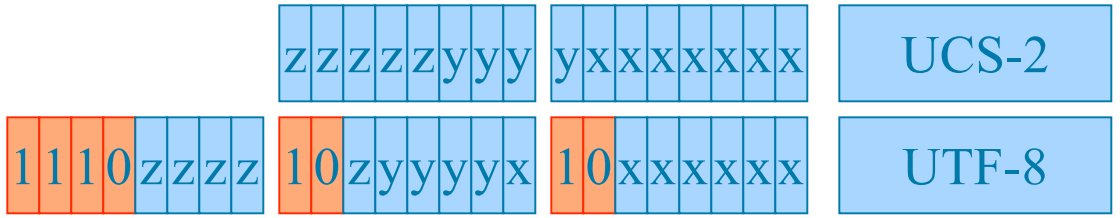
Se il primo bit è 0, si tratta di un carattere ASCII di un byte.




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



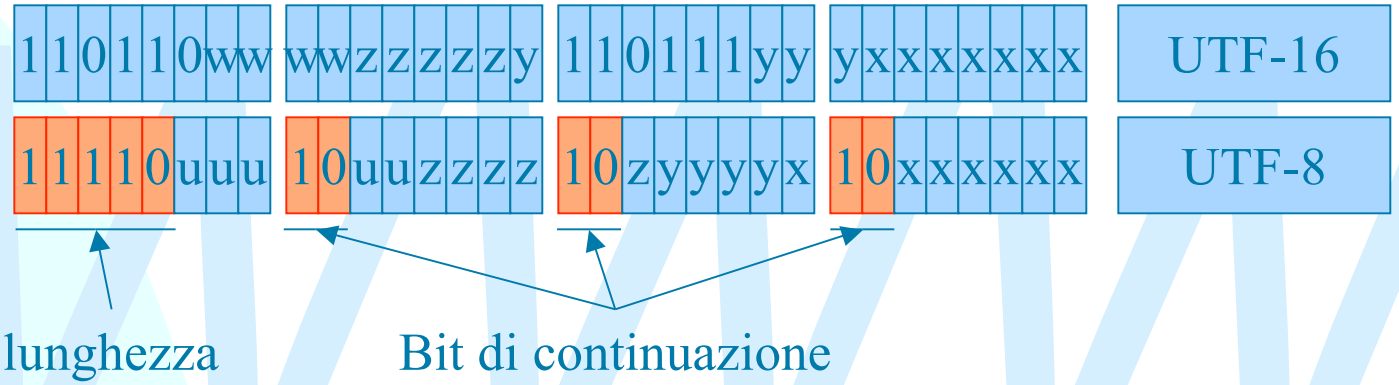

Se i primi tre bit sono 111, si tratta di un carattere appartenente ad un alfabeto ideografico.






# 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 (*bit di continuazione*) e 6 bit significativi.

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



# UTF-32

UTF-32 è una codifica a 32 bit dei caratteri Unicode. Poiché Unicode definisce un sottoinsieme dei caratteri di UCS-4 (in particolare tutti i caratteri UCS-4 non riservati),

UTF-32 è identico a UCS-4 per questi caratteri.

Non c'è nessuna ulteriore differenza se non l'esclusione di codici leciti in UCS-4 e illeciti in Unicode.



# UTF-7

Esiste anche una versione a 7 bit di Unicode, che permette di utilizzare un certo numero di canali a 7 bit (ad esempio, SMTP) per trasmettere senza problemi testi internazionali.

UTF-7 non è proposto dal consorzio Unicode, ma dall'RFC 2152 di IETF. E' pensato come formato di codifica, non di visualizzazione.

Permette di evitare la doppia trasformazione UCS-16 -> UTF-8 e UTF-8 -> ASCII 7bit, particolarmente onerosa per testi di alfabeti non-europei,

Anche UTF-7, ovviamente, è un formato a lunghezza variabile, che memorizza in maniera invariata i caratteri ASCII 7 bit (tranne il +, utilizzato come shift), e in una sequenza di 2-8 caratteri ASCII 7 bit.



# 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

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.



# 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)
- I problemi di codifica e di ordinamento dei byte



# 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, *The online edition of the Unicode standard, version 3.0*, http://www.unicode.org/uni2book/u2.html