Capitolo 47

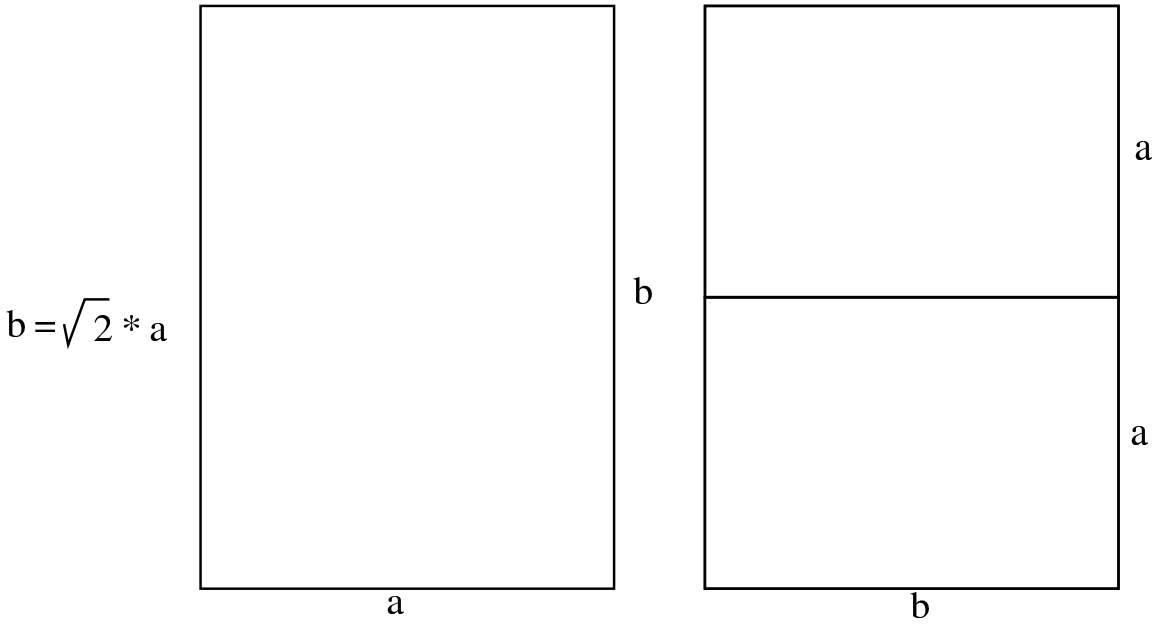
Editoria e stile

«

47.1 Formati standard della carta

Lo standard ISO 216, così come gli standard UNI 936 e DIN 476, definisce i formati di carta più comuni, secondo una logica molto semplice: i lati del foglio di carta hanno un rapporto fisso, dove il lato lungo è pari alla radice quadrata di due (circa 1,4142) per la lunghezza del lato corto (figura 47.1).

Figura 47.1. Il rapporto tra i lati di un foglio ISO 216.



Questo rapporto ha una proprietà importante, che consente al foglio di carta di essere dimezzato sul lato lungo, oppure di essere raddoppiato sul lato corto, mantenendo lo stesso rapporto tra i lati.

Lo standard ISO 216 definisce tre diverse serie di questi formati, ognuna delle quali parte da una dimensione di partenza, generando le altre dimensioni suddividendo quella precedente a metà, sul lato lungo. La serie A, ha come punto di riferimento il formato A0, corrispondente a un foglio con un'area di un metro quadro, tuttavia non si tratta del formato più grande, che è ottenuto raddoppiando due volte il formato A0, ottenendo così quattro metri quadri.

La tabella 47.2 elenca le dimensioni di tutti i formati delle tre serie, denominate A, B e C. Come si può osservare, i valori sono approssimati al millimetro, in aderenza al SI (sezione 47.3.3).

Tabella 47.2. ISO 216: formato A, B e C.

A	mm	B	mm	C	mm
4A0	1682×2378	--	--	--	--
2A0	1189×1682	--	--	--	--
A0	841×1189	B0	1000×1414	C0	917×1297
A1	594×841	B1	707×1000	C1	648×917
A2	420×594	B2	500×707	C2	458×648
A3	297×420	B3	353×500	C3	324×458
A4	210×297	B4	250×353	C4	229×324
A5	148×210	B5	176×250	C5	162×229
A6	105×148	B6	125×176	C6	114×162
A7	74×105	B7	88×125	C7	81×114
A8	52×74	B8	62×88	C8	57×81
A9	37×52	B9	44×62	C9	40×57
A10	26×37	B10	31×44	C10	28×40

47.1.1 Utilizzo pratico dei vari formati ISO 216

Tabella 47.3. Esempi di utilizzo pratico dei vari formati.

Formati	Utilizzo
A0, A1	Disegno tecnico; poster.
A2, A3	Disegno; diagrammi; tabelle di grandi dimensioni.
A4	Lettere; riviste; cataloghi; carta per stampanti comuni e per fotocopiatrici.
A5	Blocchi per appunti.
C4	Buste per il formato A4.
C5	Buste per il formato A4 piegato a metà.
C6	Buste per il formato A4 piegato due volte.
B4, A3	Giornali.

La percentuale di ingrandimento o di riduzione di un formato per ottenerne un altro, si determina facilmente, tenendo conto che si sta facendo riferimento all'ampiezza e all'altezza del foglio, non alla sua area. In pratica, riducendo un formato A4 al 50 %, si ottiene un formato A6, mentre per arrivare al formato A5 occorre usare una riduzione al 71 %. In altri termini, 71 %, ovvero 0,71, approssima la radice quadrata di 0,5. La tabella 47.4 riepiloga alcune trasformazioni tipiche, da un formato a un altro dello standard ISO 216.

Tabella 47.4. Esempi di ingrandimento e riduzione dei formati più comuni.

Trasformazione richiesta	rapporto	percentuale (approssimata)
da A***n*** a A***n***+1	$\sqrt{0{,}5}$	71 %
da B***n*** a A***n***	$\sqrt{\sqrt{0{,}5}}$	84 %
da A***n*** a B***n***	$\sqrt{\sqrt{2}}$	119 %
da B***n*** a A***n***−1	$\sqrt{\sqrt{2}}$	119 %
da A***n*** a A***n***−1	$\sqrt{2}$	141 %

La massa di un foglio di serie A, può essere determinata facilmente, sapendo che A0 ha una superficie di un metro quadro. In pratica, basta conoscere la densità superficiale della carta (la cosiddetta grammatura) che si esprime normalmente in grammi per metro quadro, dividendone opportunamente il valore: l'A***n*** ha una massa pari a 2^{-n} volte quella dell'A0. Per esempio, la massa di un foglio A4 è 2^{-4} volte quella di un A0; ovvero 1/16; se la grammatura è 80 g/m^2, la massa di un foglio A4 è 5 g.

Le dimensioni dei fogli delle tre serie ISO 216 possono essere determinate anche attraverso delle formule matematiche, come mostrato nella tabella 47.5. Si osservi che le misure che si ottengono sono

espresse in metri.

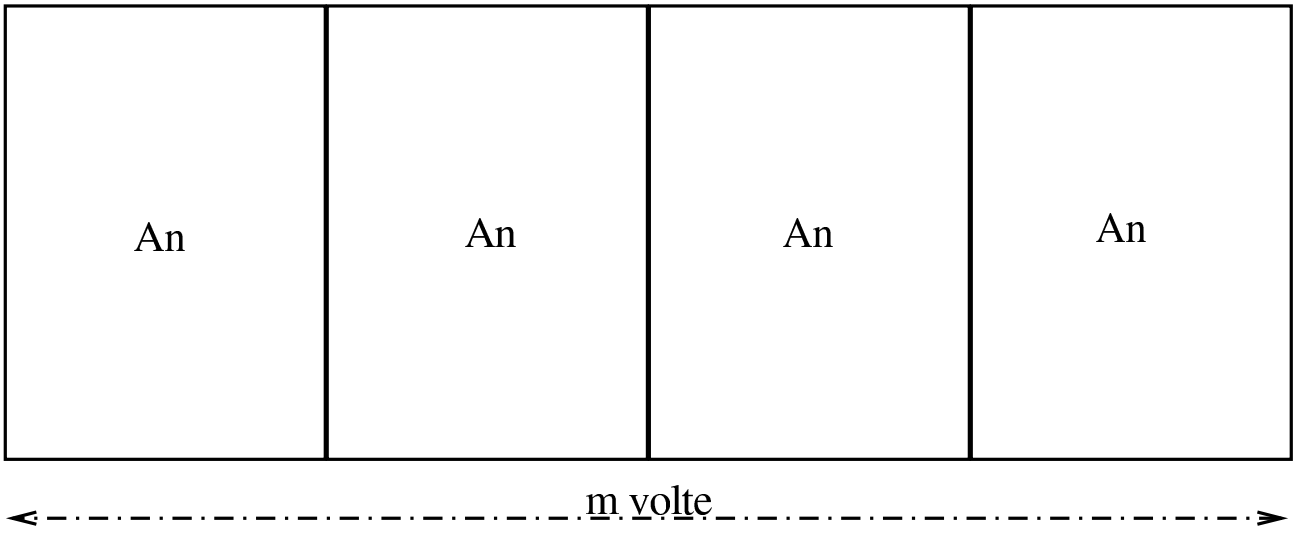
Tabella 47.5. Formule per calcolare le dimensioni della carta secondo lo standard ISO 216.

Formato	Ampiezza in metri	Altezza in metri
An	$2^{(-1/4-n/2)}$	$2^{(1/4-n/2)}$
Bn	$2^{(-n/2)}$	$2^{(1/2-n/2)}$
Cn	$2^{(-1/8-n/2)}$	$2^{(3/8-n/2)}$

47.1.2 Formati multipli

Quando non si può utilizzare un formato in cui il rapporto tra la lunghezza dei lati sia quello delle serie A, B o C comuni, si possono usare dei multipli di uno di questi formati. Come si vede nella figura 47.6, si tratta di affiancare più fogli di un certo formato, estendendo il lato corto. Questi formati estesi si indicano come Anxm, dove m rappresenta quanti fogli di tipo An affiancare. Per esempio, il formato A3 è equivalente al formato A4x2.

Figura 47.6. Formati multipli Anxm.



47.2 Nozioni elementari di tipografia

«

Prima di studiare un programma di editoria elettronica conviene conoscere almeno qualche nozione di tipografia. Studiando la natura del problema si può comprendere la ragione di alcuni comportamenti dei programmi più raffinati che rispecchiano, nella loro impostazione, la filosofia della tipografia tradizionale.

47.2.1 Carattere

«

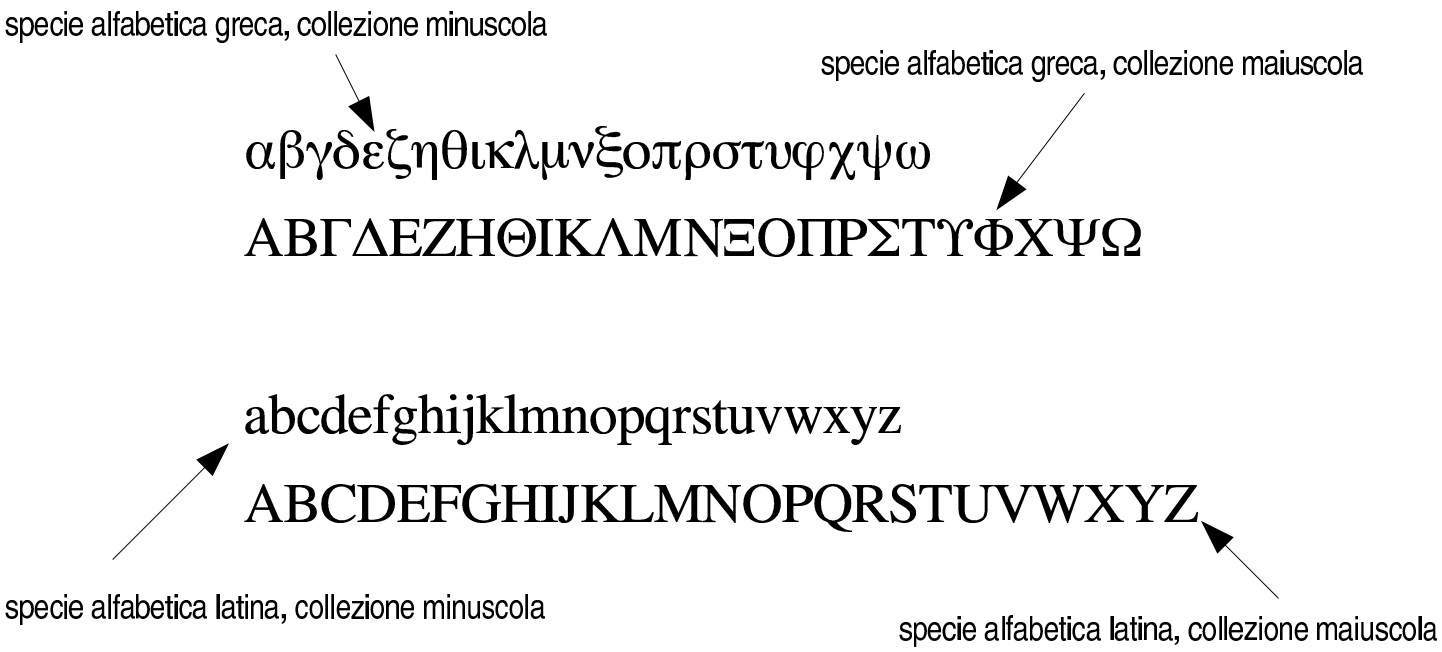
Il ***carattere*** è qualunque segno grafico utilizzato in tipografia per rappresentare le lettere, i segni di interpunzione, le cifre e altri grafemi. La conoscenza delle caratteristiche fondamentali del carattere da stampa è utile per poter comprendere il funzionamento e la logica dei programmi di composizione tipografica. Sul carattere si possono distinguere diversi aspetti, in particolare: la specie alfabetica; lo stile, o il gruppo stilistico; la serie alfabetica, o la variante di serie; la scala dimensionale.

47.2.1.1 Specie alfabetica

«

La ***specie*** è una collezione di segni di un tipo di scrittura. Per quanto riguarda l'europa occidentale, la specie alfabetica comune è quella dell'alfabeto latino. All'interno di una specie alfabetica si possono distinguere diverse collezioni alfabetiche, per esempio come nella distinzione tra lettere maiuscole e minuscole che avviene nell'alfabeto latino.

Figura 47.7. Specie e collezione.



Dalla differenza tra gli alfabeti nasce a volte la necessità di rendere un testo attraverso un alfabeto alternativo. La ***traslitterazione*** è il procedimento di traslazione da un sistema alfabetico a un altro, in modo da ricomporre un testo facendo uso di un sistema alfabetico diverso da quello originale. La traslitterazione punta a riprodurre un testo in modo che sia possibile in qualsiasi momento il procedimento inverso per riottenere il testo originale. Il caso più comune in cui si ha la necessità di utilizzare la traslitterazione è quello della citazione in cui l'originale utilizza un alfabeto esotico per il quale non si dispone del carattere tipografico. Come si può immaginare, la traslitterazione è regolata da norme internazionali.

47.2.1.2 Gruppo stilistico

Una volta definita la specie di un carattere si possono distinguere delle varianti che riguardano lo ***stile***, ovvero il disegno e il suo gusto estetico. Sull'alfabeto latino sono stati realizzati una quantità così grande di stili diversi che è difficile persino riuscire a classificarli.

In generale vi si fa riferimento attraverso il nome. Gli stili tradizionali comuni nella composizione elettronica sono: Times, Helvetica e Courier.

Figura 47.8. Stile.

carattere con grazie:

αβγδεζηθικλμνξοπρστυφχψω
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

carattere senza grazie:

αβγδεζηθικλμνξοπρστυφχψω
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

carattere con grazie a spaziatura orizzontale uniforme:

αβγδεζηθικλμνξοπρστυφχψω
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

I tre nomi citati rappresentano oggi, simbolicamente, le caratteristiche fondamentali di uno stile: la presenza o l'assenza di grazie e la proporzionalità o meno della larghezza dei segni.[1]

Le grazie sono dei piedini terminali che hanno lo scopo di abbellire il carattere e di guidare la vista durante la lettura. Il Times è il tipico stile con grazie, mentre Helvetica è il suo opposto.

I segni dei caratteri da stampa sono generalmente di larghezza diver-

sa; solo le prime forme di scrittura meccanica, come la macchina da scrivere e le prime stampanti, hanno creato la necessità di utilizzare dei simboli a larghezza uniforme. Il Courier è il rappresentante di questo tipo di stile a larghezza fissa.

In generale, uno stile riguarda esclusivamente una specie alfabetica, ma quando uno stile assume importanza e notorietà, può succedere che venga adottato anche da altre specie. Per questo si può distinguere tra Times Roman (Times New Roman), Times Greco, Times Cirillico e altri. Il primo tra quelli citati è ovviamente il Times dell'alfabeto latino.

47.2.1.3 Serie

La ***serie alfabetica***, o la variante di serie, rappresenta una distinzione all'interno di uno stile, in base alla ***forma***. Le forme comuni di uno stesso stile riguardano la pendenza, il tono e la larghezza.

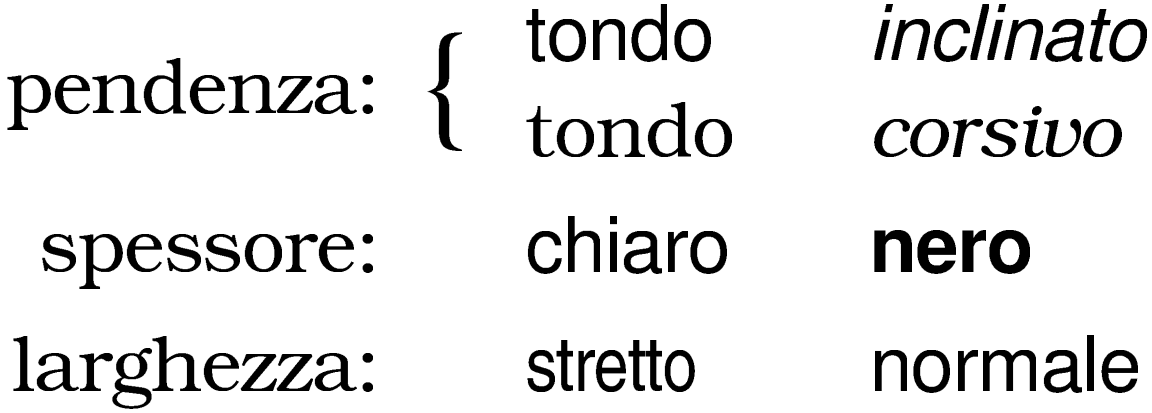
- La pendenza si riferisce all'inclinazione delle aste e si distingue generalmente tra ***tondo***, rappresentato da un carattere con aste verticali, e ***corsivo***, in cui le aste sono inclinate in avanti. Generalmente, l'aspetto dei caratteri di un corsivo, pur restando all'interno della stesso stile, è abbastanza diverso da quello del tondo. Quando si utilizza un sistema di composizione elettronico può capitare di avere a disposizione uno stile nel quale manchi il corsivo, ottenendolo però in qualche modo distorcendo il tondo. In questo caso si parla preferibilmente di carattere «inclinato» in modo volutamente generico.
- Il tono, o lo spessore, rappresenta l'intensità del carattere che si percepisce visivamente. Essendo un concetto che deriva dal-

la stampa con inchiostro nero, si distingue generalmente tra ***chiarissimo***, ***chiaro***, ***nero*** (***neretto***) e ***nerissimo***.

- La larghezza è una caratteristica di cui dispongono solo alcuni stili, ovvero li può riguardare direttamente, nel senso che uno stile per sua natura può essere «stretto» o «largo». In base alla larghezza si distinguono solitamente: lo ***strettissimo***, lo ***stretto***, il normale, il ***largo*** e il ***larghissimo***.

È bene chiarire che ogni stile può disporre o meno di varianti seriali adatte. Alcuni stili, spesso riferiti a specie alfabetiche simboliche, dispongono di una serie unica.[2]

Figura 47.9. Serie.



47.2.2 Tipometria

«

La tipometria è la misurazione degli elementi che riguardano la composizione e l'impaginazione. Le voci più importanti sono costituite dai corpi (l'altezza dei caratteri), dalla spaziatura, dall'interlinea, dalla giustezza e dalla giustificazione. In breve, il corpo è l'altezza del carattere, la spaziatura è la distanza tra una parola e l'altra in una riga, l'interlinea è lo spazio verticale aggiuntivo tra le righe, la giustezza è lo spazio orizzontale che le righe di testo hanno a

disposizione, la giustificazione è il procedimento di regolazione della spaziatura e dell'interlinea in modo da ottenere un allineamento delle righe con i margini (sia in orizzontale, sia in verticale).

47.2.2.1 Corpo, dimensioni e scala

La dimensione del carattere si misura in senso verticale e si definisce ***corpo***. Per misurare il corpo e le altre dimensioni che riguardano i caratteri si possono utilizzare diverse unità di misura, ma quando si tratta di sistemi di composizione elettronica a mezzo di software, è molto probabile che si disponga solo del pica e del punto anglo-americano:

- 1 pica = 1/6 di pollice;
- 1 punto = 1/12 di pica = 1/72 di pollice.

Per comprendere cosa sia il corpo di un carattere è bene descrivere le varie componenti dell'altezza di questo. La figura 47.10 mostra schematicamente la parola «Agglomerato» suddivisa secondo le componenti verticali della dimensione del carattere.

Figura 47.10. Le dimensioni del carattere.

Il carattere si appoggia su una linea che rappresenta la base della «parte mediana»; le lettere come la «l» si alzano occupando anche la «parte ascendente»; altre, come la «g», si allungano in basso a occupare la «parte discendente». Il corpo del carattere include anche

uno spazio aggiuntivo: la «spalla». Si distingue una spalla superiore, costituita da uno spazio minimo sopra la parte ascendente, e la spalla inferiore che si trova al di sotto della parte discendente (nella figura la spalla è molto grande, in proporzione, rispetto alla realtà).

La distanza tra la base di una riga (la base della parte mediana) e la base di quella successiva dovrebbe essere superiore o al minimo uguale alla grandezza del corpo. Quando questa distanza è superiore, lo spazio aggiuntivo è l'***interlinea***. Con i sistemi di composizione elettronica per mezzo di software, si misura generalmente lo spazio tra le basi delle righe ed è ammissibile anche l'utilizzo di distanze inferiori all'altezza del carattere, ottenendo in pratica una sovrapposizione della parte mediana inferiore di una riga con la parte mediana superiore di quella successiva.

La rappresentazione di un carattere con un corpo di una data dimensione dipende dalla disponibilità di questo. Con i sistemi tipografici meccanici è necessario disporre di una serie di caratteri mobili differenti, distinti in base a una scala. Con i sistemi di composizione elettronica via software si possono trovare dei caratteri riproducibili in qualsiasi corpo, eventualmente generando dei file opportuni per la scala richiesta. Tuttavia, in presenza di dimensioni particolarmente piccole si rischia di perdere dei dettagli importanti dei segni che compongono lo stile utilizzato e, di conseguenza, potrebbe essere preferibile l'utilizzo di una variante dello stile che sia più adatta alle dimensioni ridotte.

47.2.2.2 Giustezza, spaziatura e giustificazione orizzontale

La giustezza è lo spazio orizzontale a disposizione delle righe di testo; in altri termini, è la larghezza della colonna all'interno della quale si può distribuire il testo. La spaziatura è lo spazio tra la fine di una parola e l'inizio di quella successiva.

Nei testi in italiano, la spaziatura è uniforme, senza eccezioni, a differenza della tradizione tipografica di altri paesi. Per esempio, la spaziatura dopo un punto fermo è esattamente uguale a quella di qualunque altra situazione. Quando si utilizza il sistema di composizione TeX per scrivere un testo in italiano, si dovrebbe inserire il comando '`\frenchspacing`' per evitare anomalie nella spaziatura.

Quando si vuole ottenere un allineamento del testo all'inizio e alla fine della giustezza, si parla di giustificazione (orizzontale). Per ottenerla, è necessario che la spaziatura sia adattata in modo da arrivare a questo risultato. La giustificazione orizzontale è solo una delle scelte stilistiche che il tipografo ha a disposizione: non si tratta di una convenzione obbligatoria.

47.2.2.3 Giustificazione verticale

Come nel caso della giustificazione orizzontale, ci può essere la necessità o l'opportunità di adattare l'interlinea in modo da riempire completamente le pagine. Ciò si ottiene attraverso la giustificazione verticale.

47.2.3 Il carattere nel software di composizione

«

Utilizzando i programmi di composizione tipografica si è costretti generalmente a fare i conti con la terminologia dei paesi di lingua inglese e con altri problemi legati alla rappresentazione simbolica dei segni all'interno del software. La tradizione tipografica di questi ha generato dei termini che non sono perfettamente traducibili con concetti della tradizione italiana, per cui si utilizzano alcuni termini di origine anglofona, eventualmente tradotti in modo letterale.

47.2.3.1 Terminologia

«

In inglese si utilizza normalmente il termine *font* per fare riferimento al carattere tipografico. Spesso non si traduce questo termine in qualcosa che riguardi la tradizione tipografica italiana, mantenendo piuttosto il termine inglese invariato; tuttavia, alle volte viene utilizzata la forma: ***fonte***.

Se il contesto non richiede un'aderenza perfetta con il termine originale inglese, si possono usare forme espressive più comprensibili, come «carattere», «tipo di carattere», «carattere tipografico» o «carattere da stampa».

47.2.3.2 Caratteristiche di una carattere tipografico elettronico

«

Il carattere tipografico usato nel software applicativo di composizione, ha una serie di caratteristiche, alcune delle quali sono fondamentali.

- ***foundry*, *fonderia***

  La fonderia è il produttore del carattere tipografico, cioè chi ha creato la tipizzazione,[3] pur senza esserne il disegnatore. Per fare un esempio comune, Adobe è la fonderia dello stile Times New Roman.

- ***family*, *famiglia***

  La famiglia del carattere, inteso come traduzione del termine *font family*, corrisponde simultaneamente alla specie e allo stile del carattere. In altri termini, rappresenta sia la specie alfabetica, sia lo stile. Per fare un esempio, la famiglia Times New Roman è un carattere di specie latina e di stile Times.

- All'interno di una famiglia si distinguono normalmente le serie riferite alla forma: spessore (*weight*), inclinazione (*slant*) e larghezza (*set*, o *width*).

- ***codifica***

  La codifica rappresenta l'elemento nuovo più importante nelle caratteristiche di un carattere tipografico per l'elaborazione via software. Il problema viene descritto nella prossima sezione.

47.2.3.3 Codifica

L'utilizzo dei caratteri con i sistemi di composizione basati sul software richiede un abbinamento tra segni e simboli binari. Questo abbinamento è definito dalla codifica. Il problema si può intendere meglio se si pensa a un programma a composizione differita.

In questi casi si parte da un file sorgente, scritto probabilmente secondo la codifica UTF-8, con il quale il programma deve comporre il risultato, utilizzando i caratteri a disposizione.

Il carattere tipografico utilizzato dal programma di composizione è contenuto normalmente all'interno di file, da cui questo programma estrae le informazioni necessarie attraverso un riferimento dato da un codice numerico. In condizioni normali, il programma di composizione fa riferimento al simbolo binario utilizzato nel sorgente per ottenere il segno corrispondente all'interno del carattere tipografico utilizzato (eventualmente attraverso una qualche traslazione). In pratica, alla lettera «A» nel sorgente dovrebbe corrispondere la lettera «A» del carattere tipografico che si sta utilizzando, ma se il carattere tipografico è organizzata in modo differente, si potrebbe ottenere qualcosa di diverso. Questo problema si avverte di solito quando si utilizza una famiglia di caratteri che fa riferimento a una specie simbolica, o comunque a un alfabeto che non ha alcuna corrispondenza con la codifica utilizzata nel sorgente. In questi casi, di solito, per rappresentare i segni si può fare uso di comandi speciali interpretati opportunamente dal programma di composizione.

Un programma di composizione potrebbe disporre di caratteri tipografici che hanno solo una corrispondenza parziale con la codifica utilizzata per scrivere il sorgente, per esempio, potrebbero mancare alcuni segni che vengono messi a disposizione attraverso altre specie.

Il problema viene riproposto nella sezione 47.6.

47.2.4 Problemi legati ai caratteri

«

Nelle origini della tipografia, molti caratteri mobili rappresentavano l'unione di più lettere o altri segni in logotipo (cioè l'unione in un simbolo unico). L'unione di questi derivava da delle consuetudini

stilistiche o dalla forma dei segni adiacenti che per qualche motivo potevano richiedere un avvicinamento o un adattamento.

Il ***legato*** (in inglese *ligature*) è l'unione di due o più segni per motivi storici o estetici; i più comuni sono le sequenze «fi», «fl» e «ffi», dove le lettere vengono avvicinate in modo particolare fino a unirsi o a inglobarsi. Alcune forme di legato si sono tradotte in segni indipendenti, come nel caso di «AE» che si è trasformato in «Æ», «sz» che nella lingua tedesca è ormai «ß», «et» (latino) che è divenuto «&», ovvero l'attuale e-commerciale, e anche «ad» (latino), che nella lingua inglese è diventato «@» (*at*).

L'avvicinamento delle lettere, è motivato dalla forma di queste, per evitare il formarsi di vuoti visivi che potrebbero creare difficoltà alla lettura. I casi più comuni sono le sequenze «AV», «AT», «AY».

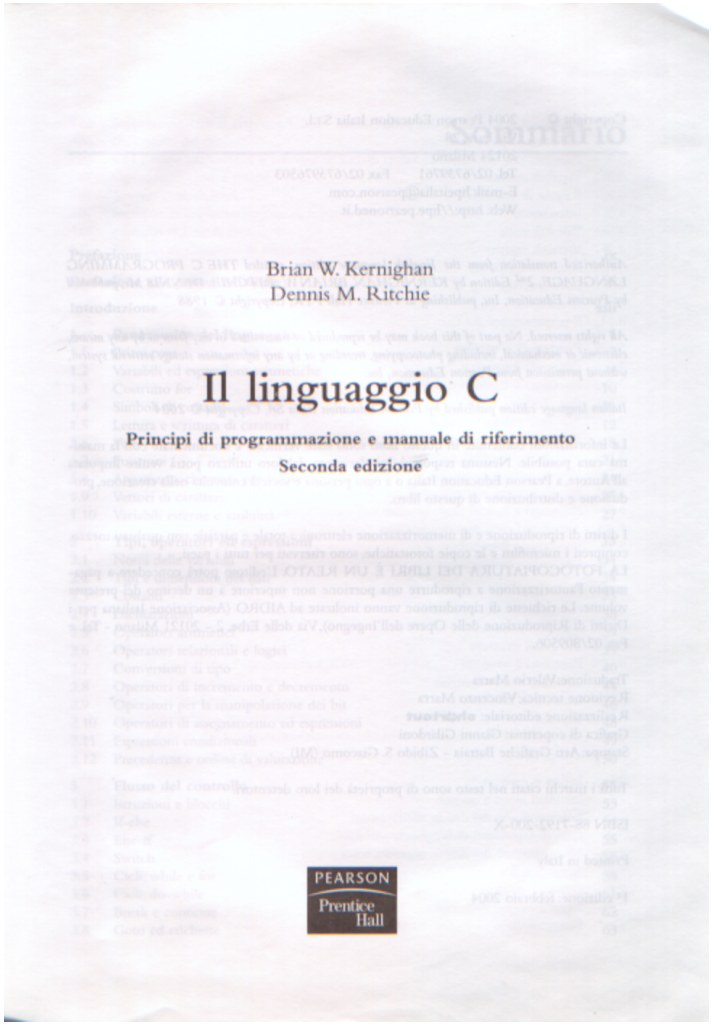
47.2.5 Il libro

«

Alcune componenti del libro, nella sua forma cartacea, hanno dei nomi particolari, che spesso si ignorano. Le figure successive descrivono sommariamente tali componenti, utilizzando come esempio il libro *Il linguaggio C* di Brian W. Kernigham e Dennis M. Ritchie, edizione Pearson, 2004.

Figura 47.11. Ciò che appare esternamente a un libro è: la ***prima di copertina*** (immagine a sinistra), o semplicemente «copertina», costituente la prima pagina assoluta; la ***quarta di copertina*** (immagine a destra) è l'ultima pagina assoluta; il ***dorso***, o ***costola*** (immagine in basso), è ciò che si trova in corrispondenza della rilegatura dei fogli, con le informazioni essenziali del libro, utili per identificarlo quando viene riposto in uno scaffale.

Figura 47.12. Immagine a sinistra: il ***frontespizio*** è una delle prime pagine del libro, la quale riprende le informazioni della copertina. Immagine a destra: la pagina che contiene le informazioni sulla pubblicazione (i diritti dell'opera, le sue varie edizioni, ecc.) è il ***colofone*** o *colophon* (la dizione *colophon* viene dal latino, ma il termine originario è *kolophon* e viene dal greco). Può trovarsi alla fine dell'opera oppure tra le primissime pagine.

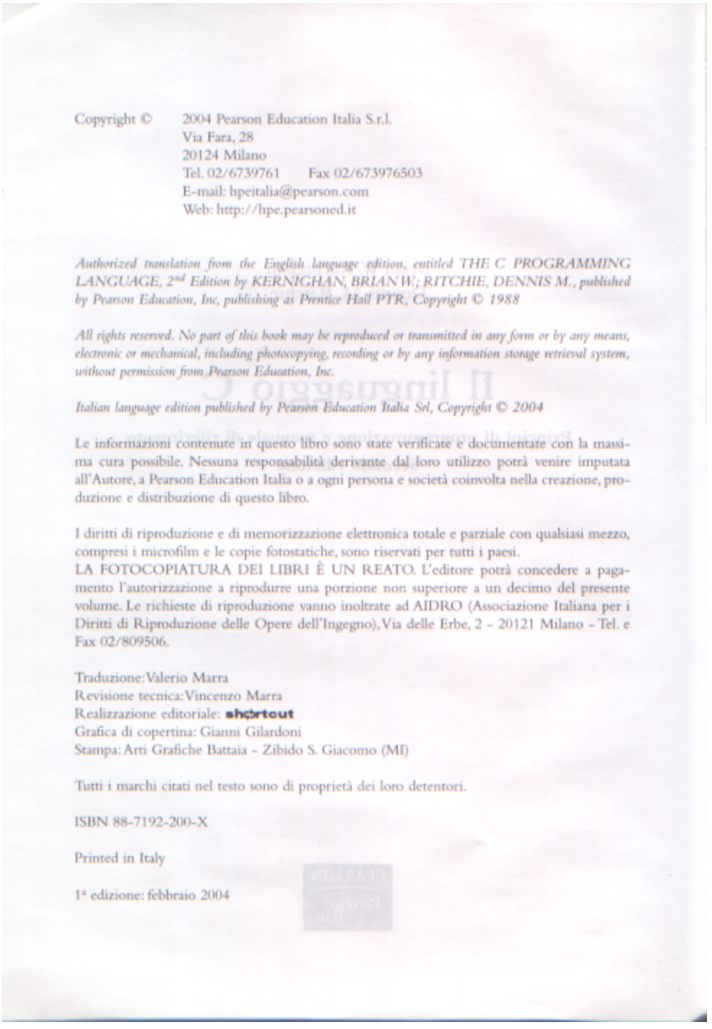


Brian W. Kernighan
Dennis M. Ritchie

Il linguaggio C

Principi di programmazione e manuale di riferimento
Seconda edizione

PEARSON
Prentice Hall



Via Fara, 28
20124 Milano
Tel. 02/6739761 Fax 02/673976503
E-mail: hpeitalia@pearson.com
Web: http://hpe.pearsoned.it

Authorized translation from the English language edition, entitled THE C PROGRAMMING LANGUAGE, 2nd Edition by KERNIGHAN, BRIAN W.; RITCHIE, DENNIS M., published by Pearson Education, Inc, publishing as Prentice Hall PTR, Copyright © 1988

Traduzione: Valerio Marra
Revisione tecnica: Vincenzo Marra
Realizzazione editoriale: shortcut
Grafica di copertina: Gianni Gilardoni
Stampa: Arti Grafiche Battaia - Zibido S. Giacomo (MI)

ISBN 88-7192-200-X

Printed in Italy

1ª edizione: febbraio 2004

Oltre agli esempi che appaiono nelle figure, vanno considerati anche la ***seconda di copertina***, che è la parte interna della prima di copertina, così come la ***terza di copertina***, che è la parte interna della quarta di copertina. Nei libri comuni, la seconda e la terza di coperti-

na sono bianche, o comunque prive di contenuti, mentre nelle riviste diventano posizioni significative per la collocazione della pubblicità.

Lo specchietto successivo consente di abbinare alcuni dei termini descritti alla forma inglese corrispondente:

prima di copertina	*front cover page*
seconda di copertina	*inside front cover page*
terza di copertina	*inside back cover page*
quarta di copertina	*back cover page*

47.3 Stile letterario[4]

«

Il concetto di ***stile letterario*** potrebbe essere espresso semplicemente spiegando l'esigenza di realizzare un documento ***uniforme***: sia dal punto di vista visivo, sia dal punto di vista espressivo. Questo coinvolge quindi l'aspetto grammaticale (ortografia, sintassi, lessico, ecc.) e l'aspetto tipografico (impaginazione, tipi di carattere, dimensione, ecc.) o artistico.

L'esigenza di un'uniformità visiva deriva dal piacere e dal rilassamento che può dare al lettore un documento impaginato e strutturato in un modo ordinato e chiaro, per la facilità nella lettura che ne deriva. Nello stesso modo è importante l'uniformità grammaticale, cosa particolarmente delicata in una lingua come quella italiana in cui sono consentite molte variazioni, data la ricchezza linguistico-culturale delle varie regioni.

Il problema dell'uniformità stilistica si accentua quando si deve collaborare alla realizzazione di un progetto letterario. L'uniformità non è più solo un fatto di coerenza personale, ma di coerenza complessiva di tutto il gruppo.

47.3.1 Regole di composizione del testo

Il modo migliore per definire uno stile grammaticale è lo studio su un testo di grammatica. Nelle sezioni successive si raccolgono solo alcuni punti essenziali che non possono essere ignorati.

47.3.1.1 Punteggiatura e spaziatura

La punteggiatura si compone di quei simboli che consentono di separare le parole e di delimitare le frasi.

- Ogni parola è separata da un solo spazio.

  Tipograficamente, lo spazio è una separazione di ampiezza non definita, spesso ampliato o compresso, per ottenere un allineamento del testo, sia a sinistra, sia a destra. Un autore non deve pensare a queste cose quando scrive la propria opera; si deve limitare a spaziare le parole con un solo carattere spazio.[5]

  La dattilografia insegnava a ottenere testi allineati a sinistra e a destra con l'inserzione opportuna di spazi aggiuntivi, vicino alle parole composte da poche lettere (congiunzioni, articoli, ecc.). Questo tipo di tecnica è ormai da abbandonare, lasciando semmai che siano i programmi di composizione a prendersi cura di tali problemi, anche quando il risultato finale deve essere un file di testo puro e semplice.

  I programmi di composizione più evoluti facilitano il compito dello scrittore eliminando gli spazi superflui, per cui con questi non c'è l'esigenza di porre attenzione alla dimensione delle spaziature.[6]

- Il simbolo di punteggiatura normale è attaccato alla parola che precede ed è separato con uno spazio dalla parola che segue.

Si tratta di: punto, virgola, due punti, punto e virgola, punto interrogativo e punto esclamativo.

Alle volte, l'autore di documenti tecnici di informatica si lascia confondere dall'uso che si fa di tali simboli in un particolare linguaggio di programmazione o in altri ambiti analoghi. È chiaro, per esempio, che se si deve indicare un'estensione di un file, come «.sgml», non si può rispettare tale regola, in quanto il punto che precede quell'estensione non rappresenta un simbolo di punteggiatura del testo.

- Le parentesi sono attaccate al testo che racchiudono e, rispetto alla punteggiatura esterna, si comportano come un'unica parola.

  La parentesi di apertura è separata con uno spazio dalla parola che precede, mentre quella di chiusura è separata con uno spazio dalla parola che segue. I simboli di punteggiatura normale che dovessero seguire una parentesi chiusa vanno attaccati a questa ultima.

  Nella lingua italiana non è consentito racchiudere all'interno di parentesi un periodo terminante con il punto fermo (al contrario di quanto avvenga con la lingua inglese).

- Il testo riportato tra virgolette si comporta come quello racchiuso tra parentesi.

  La lingua italiana prevede l'uso di virgolette uncinate (in basso), virgolette elevate doppie e singole. Secondo la grammatica, le virgolette uncinate, o virgolette basse, sono da preferire. Tuttavia, dal momento che le virgolette elevate possono essere ottenute anche utilizzando soltanto il codice ASCII tradizionale a 7 bit, molti autori preferiscono accontentarsi e utilizzare solo quelle elevate.[7]

- Il trattino di unione è corto e unito alle parole da collegare.

  Si usa per unire insieme due parole in modo da formare una parola composta. I programmi di composizione tendono a considerare un trattino singolo come un trattino corto, proprio per questo scopo.

- La lineetta, o trattino lungo, serve per introdurre un discorso diretto, oppure un inciso.

  Il trattino utilizzato per delimitare un discorso diretto, viene usato normalmente solo in apertura. Può apparire anche un trattino in chiusura quando al discorso diretto segue un commento. Se il trattino si usa per delimitare un inciso, si usa per aprirlo e solitamente anche per chiuderlo, come se si trattasse di parentesi.

  Generalmente, il trattino lungo è preceduto e seguito da uno spazio; davanti al trattino di chiusura vanno collocati il punto interrogativo, il punto esclamativo e i puntini, mentre per gli altri simboli di punteggiatura non esiste una convenzione precisa.[8]

47.3.1.2 Utilizzo dei simboli di interpunzione

L'uso della punteggiatura nella lingua italiana è definito da regole molto vaghe che si prestano a facili eccezioni di ogni tipo. Qui si elencano solo alcuni concetti fondamentali.

,	La virgola è un segno di interpunzione che collega due segmenti di testo separati da una pausa debole.
;	Il punto e virgola è un segno di interpunzione che si colloca a metà strada tra la virgola e il punto. Non segna la chiusura di un periodo.

:	I due punti sono un simbolo di interpunzione ***esplicativo***. Collegano due segmenti di testo separati dal punto di vista sintattico, in cui la seconda parte, quella che segue il simbolo, elenca, chiarisce o dimostra il concetto espresso nella prima parte.
.	Il punto fermo è un segno di interpunzione che collega due segmenti di testo separati da un pausa forte. Generalmente segna la conclusione di un periodo. La parola successiva al punto ha l'iniziale maiuscola.
!	Il punto esclamativo indica generalmente la conclusione di un'esclamazione affermativa. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.
?	Il punto di domanda indica un tono interrogativo alla fine di una frase. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.
...	I punti di sospensione sono in numero fisso di tre e indicano che il discorso non viene portato a conclusione. Generalmente, sono uniti alla parola o al segno di interpunzione che li precede, oppure distanziati, a seconda che siano solo una sospensione oppure indichino l'omissione di un nome o di un'altra parola. Se si trovano alla fine di un periodo, dove andrebbe collocato un punto, questo non viene aggiunto e la frase successiva inizia con la maiuscola. Nello stesso modo, se si trovano alla fine di un'abbreviazione che termina con un punto, questo punto viene assorbito.
ecc**.**	Il punto di abbreviazione, quando si trova alla fine di un periodo, conclude da solo anche il periodo stesso, ed è seguito da iniziale maiuscola.

( )	Le parentesi, generalmente tonde, servono per delimitare un inciso, come un commento, una nota dello scrivente, un chiarimento, ecc. Generalmente, i commenti del redattore o del traduttore sono terminati, entro l'ambito delle parentesi, con le sigle NdR (nota del redattore) e NdT (nota del traduttore).

47.3.1.3 Accenti e troncamenti

«

Nella lingua italiana scritta, l'uso degli accenti è un fatto puramente convenzionale. Ciò significa che l'accento non indica necessariamente il suono che ha effettivamente la lettera accentata, ma solo la sua rappresentazione consueta (più avanti, nella sezione 47.3.1.4 è riportato il testo originale della norma UNI 6015 sul «segnaccento obbligatorio»).[9]

- Nella lingua scritta è prevista (ed è obbligatoria) solo l'accentazione delle vocali finali delle parole nelle quali il tono della voce si rafforza sull'ultima sillaba (accento grafico).

  È possibile l'uso dell'accento per le vocali interne quando ciò serva per togliere ambiguità tra termini omografi (scritti nello stesso modo) che abbiano significati differenti. Generalmente, questa ambiguità è risolta dal contesto e raramente si incontra la necessità di utilizzare accenti interni.

- Si utilizza comunemente solo l'accento grave (**àèìòù**), con l'eccezione della vocale «e» che può avere l'accento acuto (**é**).

- Vogliono l'accento acuto le parole terminanti in **ché** (perché, poiché, ecc.), oltre a **né** (congiunzione) e **sé** (pronome tonico). In

particolare, **sé** viene scritto generalmente senza accento quando è seguito da **stesso**, anche se la grammatica non lo richiede.

- Vogliono l'accento alcuni monosillabi contenenti due vocali: **ciò**, **già**, **giù**, **più** e **può**.
- Vogliono l'accento i monosillabi che senza potrebbero avere un significato differente. La tabella 47.15 mostra l'elenco dei monosillabi accentati più importanti.
- Non vogliono l'accento alcuni monosillabi tra cui: **qui**, **qua**, **sto** e **sta**.
- Solo alcune parole tronche richiedono la segnalazione di tale troncamento con l'apostrofo finale. In particolare: **po'** (poco), **mo'** (modo), **ca'** (casa) e alcuni imperativi.
- L'accento circonflesso (^) non si usa più. Serviva per i nomi terminanti in **–io** che al plurale terminerebbero in **–ii** (per esempio: armadio, armadii). Attualmente, si tende a usare questi plurali con una sola **–i** finale, a parte i casi in cui ciò genera ambiguità (**assassino**, **assassini**; **assassinio**, **assassinii**).

Tabella 47.15. Elenco dei monosillabi accentati più importanti e dei loro equivalenti (omografi) non accentati.

dà	indicativo di dare (dà valore)	da	preposizione (da voi)
è	verbo	e	congiunzione
là	avverbio (resta là)	la	articolo
lì	avverbio (vado lì)	li	pronome
né	congiunzione (né questo né quello)	ne	pronome (ne voglio ancora)
sé	pronome tonico (pieno di sé)	se	pronome atono o congiunzione

sì	avverbio (dice di sì)	si	pronome

Alle volte, l'uso delle vocali accentate può creare problemi tecnici, dovuti alla loro mancanza nell'insieme di caratteri a disposizione. Nelle circostanze in cui non è possibile scrivere direttamente con lettere accentate (per esempio quando si dispone di un sistema configurato male, o la tastiera non dispone dei simboli necessari), occorre utilizzare delle tecniche di rappresentazione che dipendono dal programma utilizzato per la composizione.

Con l'introduzione dell'insieme di caratteri universale, che nei sistemi Unix si attua attraverso la codifica UTF-8, il problema della rappresentazione dei caratteri con lettere accentate, viene meno, qualunque sia la lingua da utilizzare.

Riquadro 47.16. Vocali accentate attraverso l'uso di macro SGML e XML.

Vocale accentata	Macro corrispondente	Vocale accentata	Macro corrispondente
à	à	À	À
è	è	È	È
ì	ì	Ì	Ì
ò	ò	Ò	Ò
ù	ù	Ù	Ù
é	é	É	É

Riquadro 47.17. Vocali accentate per TeX.

Vocale accentata	Codice TeX corrispondente	Vocale accentata	Codice TeX corrispondente
à	\‘a	À	\‘A
è	\‘e	È	\‘E
ì	\‘{\i}	Ì	\‘I
ò	\‘o	Ò	\‘O
ù	\‘u	Ù	\‘U
é	\’e	É	\’E

Riquadro 47.18. Trucco per rappresentare le vocali accentate quando non si può fare altrimenti.

Vocale accentata	Vocale apostrofata corrispondente	Vocale accentata	Vocale apostrofata corrispondente
à	a‘	À	A‘
è	e‘	È	E‘
ì	i‘	Ì	I‘
ò	o‘	Ò	O‘
ù	u‘	Ù	U‘
é	e’	É	E’

47.3.1.4 Segnaccento obbligatorio (UNI 6015)

«

Quello che segue è la norma UNI 6015 sull’uso degli accenti. Il testo è stato ottenuto da *Grafica; scienza, tecnologia e arte della stampa e della comunicazione*, *Preparazione del manoscritto* http://www.apenet.it/ .

Segnaccento obbligatorio nell’ortografia della lingua italiana (Uni 601567):

1. *Scopo*

La presente unificazione ha lo scopo di stabilire le regole ortografiche per il segnaccento nei testi stampati in lingua italiana, quando esso sia obbligatorio.

2. *Definizione*

2.1 Il segnaccento (o segno d'accento, o accento scritto) serve a indicare esplicitamente la vocale tonica, per esempio: *andrà*, *colpì*, *temé*, *virtù*.

2.2. Il segnaccento può essere grave (‘ ‵’) o acuto (‘ ′ ’).

3. *Uso*

Il segnaccento è obbligatorio nei casi seguenti:

3.1. Su alcuni monosillabi, per distinguerli da altri monosillabi che si scrivono con le stesse lettere ma senza accento:

ché («poiché», congiunzione causale) per distinguerlo da *che* (congiunzione in ogni altro senso, o pronome);

dà (indicativo presente di dare) per distinguerlo da *da* (preposizione) e *da'* (imperativo di dare);

dì («giorno») per distinguerlo da *di* (preposizione) e *di'* (imperativo di dire);

è (verbo) per distinguerlo da *e* (congiunzione);

là (avverbio) per distinguerlo da *la* (articolo, pronome, nota musicale);

lì (avverbio) per distinguerlo da *li* (articolo, pronome);

né (congiunzione) per distinguerlo da *ne* (pronome, avverbio);

sé (pronome tonico) per distinguerlo da *se* (congiunzione, pronome atono);

sì («così», o affermazione) per distinguerlo da *si* (pronome, nota musicale);

té (pianta, bevanda) per distinguerlo da *te* (pronome).

3.2. Sui monosillabi: *chiù*, *ciò*, *diè*, *fé*, *già*, *giù*, *piè*, *più*, *può*, *scià*.

3.3. Su tutte le parole polisillabe su cui la posa della voce cade sulla vocale che è alla fine della parola, per esempio: *pietà*, *lunedì*, *farò*, *autogrù*.

4. *Forma*

4.1. Il segnaccento, nei casi in cui è obbligatorio, è sempre grave sulle vocali: *a*, *i*, *o*, *u*.

4.2. Sulla *e*, il segnaccento obbligatorio è grave se la vocale è aperta, è acuto se la vocale è chiusa:

- è sempre grave sulle parole seguenti:

ahimè e *ohimè*, *caffè*, *canapè*, *cioè*, *coccodè*, *diè* e *gilè*, *lacchè*, *piè*, *tè*; inoltre sulla maggior parte dei francesismi adattati, come *bebè*, *cabarè*, *purè*, ecc. e sulla maggior parte dei nomi propri, come *Giosuè*, *Mosè*, *Noè*, *Salomè*, *Tigrè;*

- è acuto sulle parole seguenti:

ché («poiché») e i composti di *che* (*affinché*, *macché*, *perché*, ecc.), *fé* e i composti *affé*, *autodafé*, i composti di *re* e di *tre* (*viceré*, *ventitré*), i passati remoti (*credé*, *temé*, ecc., escluso *diè*), le parole *mercé*, *né*, *scimpanzé*, *sé*, *testé*.

4.3. Anche per la *o* si possono distinguere i due timbri (aperto o chiuso) con i due accenti (grave ed acuto) ma solo in casi in cui l'accento è facoltativo, per esempio: *còlto* (participio passato di *cogliere*), e *cólto* («istruito»).

47.3.1.5 Uso della «d» eufonica

Le congiunzioni **e**, **o** e la preposizione **a**, consentono l'aggiunta di una **d** eufonica, per facilitarne la pronuncia quando la parola che segue inizia per vocale. Si tratta di una possibilità e non di una regola; di questa **d** si potrebbe benissimo fare a meno.

Ognuno tende a usare questa **d** eufonica in modo differente, a seconda della propria cadenza personale, che ne può richiedere o meno la presenza. Quando si scrive, bisognerebbe mantenere lo stesso stile, anche sotto questo aspetto, quindi ognuno deve stabilire e seguire un proprio modo.

Esiste tuttavia un suggerimento che punta all'uso moderato di queste **d** eufoniche: usare la **d** solo quando la vocale iniziale della parola successiva è la stessa; e non usarla nemmeno quando, pur essendoci la stessa vocale iniziale nella parola successiva, ci sia subito dopo una **d** che possa complicare la pronuncia.

47.3.1.6 Elisione davanti alla lettera «h»

In linea di massima, l'articolo che si mette davanti a un termine che inizia con la lettera **h**, è quello che si userebbe pronunciando quella parola come se iniziasse per vocale. Secondo questo principio, va usata l'elisione, così come si fa con i termini che iniziano per vocale, senza alcuna «h» anteriore. Per esempio: l'harem; l'hotel; l'host.

Tuttavia, quando si tratta di un termine che, proveniendo da un'altra lingua, non è ancora diventato di uso comune e nella lingua originale si pronuncia con la lettera «h» iniziale aspirata, si preferisce evitare l'elisione.

47.3.1.7 Uso delle maiuscole

«

L'iniziale maiuscola si utilizza all'inizio del periodo e per evidenziare i nomi propri. La lingua italiana fa un uso diverso delle maiuscole rispetto ad altre lingue, limitandole al minimo indispensabile; pertanto, nelle situazioni di dubbio, è meglio utilizzare le lettere minuscole.

47.3.1.8 Plurali

«

Ci sono alcuni aspetti del plurale nella lingua italiana che vale la pena di annotare. In particolare, nel caso di chi deve utilizzare anche termini stranieri, si pone il problema di decidere se questi siano invariabili o meno. A questo proposito, esistono due regolette semplici e pratiche:

- le parole terminanti per consonante sono invariate al plurale;
- i termini di provenienza straniera non ancora assimilati sono invariati al plurale.

In particolare, per quanto riguarda la seconda, la logica è che non si può applicare un plurale secondo le regole di una lingua straniera mentre si usa l'italiano. Inoltre, dato che nella maggior parte dei casi si tratta di termini inglesi, che nella loro lingua prenderebbero quasi sempre una terminazione in **–s** al plurale, diventerebbe anche difficile la loro pronuncia in italiano.

47.3.1.9 Interfacce o interfaccie?

Esiste una regoletta che permette di stabilire facilmente come debba essere ottenuto il plurale delle parole che terminano in **–cia** e **–gia**: la **i** rimane se la **c** e la **g** sono precedute da vocale, oppure se la **i** viene pronunciata con accento, mentre viene eliminata se queste consonanti sono precedute da un'altra consonante.

Quindi si ha: **camicia**, **camicie** e **interfaccia**, **interfacce**; **ciliegia**, **ciliegie** e **spiaggia**, **spiagge**; **energia**, **energie**.

47.3.1.10 Elenchi

Gli elementi puntati, o numerati, possono essere composti da elementi brevi, oppure da interi periodi. Se tutti gli elementi sono brevi:

- l'elenco deve essere introdotto da una frase terminante con due punti;
- ogni elemento deve essere terminato con un punto e virgola, a eccezione dell'ultimo che termina normalmente con un punto.

La descrizione appena fatta mostra un esempio di elenco del genere. Se anche uno solo degli elementi è troppo lungo, è bene trasformare tutti gli elementi in periodi terminati da un punto. In tal caso, se l'elenco viene introdotto da una frase, anch'essa termina con un punto.

Ci possono essere situazioni in cui queste indicazioni non sono applicabili: come sempre è necessario affidarsi al buon senso.

47.3.1.11 Citazioni

«

Le citazioni, cioè le frasi o i brani riprodotti letteralmente da altri documenti, devono apparire distinte chiaramente dal testo normale. Si usano normalmente queste convenzioni:

- quando la citazione è incorporata nel testo viene delimitata attraverso le virgolette, oppure utilizzando il corsivo se la citazione è particolarmente breve;
- le citazioni incluse in un'altra citazione già virgolettata si evidenziano attraverso l'uso di un altro tipo di virgolette, cominciando da quelle uncinate («»), utilizzando poi quelle elevate doppie (‘“’”) e terminando con quelle singole (‘’);
- quando la citazione è molto lunga e occupa diversi capoversi, conviene utilizzare un corpo minore o un altro espediente tipografico per distinguerla dal testo normale, come con l'uso di rientri differenti;
- quando la citazione è lunga e non si vogliono utilizzare altri espedienti per evidenziarla, si utilizzano le virgolette, ripetendo quelle di apertura all'inizio di ogni capoverso;
- all'interno delle citazioni possono apparire dei commenti o chiarimenti inseriti da chi scrive, delimitandoli attraverso l'uso di parentesi quadre;
- all'interno delle citazioni vanno indicate le omissioni, le quali possono essere segnalate attraverso l'uso dei puntini di sospensione racchiusi tra parentesi quadre (come per i commenti);

- quando si fanno delle omissioni nella citazione all'inizio o alla fine del brano, è preferibile l'uso dei puntini di sospensione senza che questi siano racchiusi tra parentesi quadre; all'inizio i puntini di sospensione sono staccati dalla prima parola, mentre alla fine sono attaccati all'ultima.

47.3.1.12 Acquisizione di termini stranieri

Quando si decide di utilizzare un termine straniero nel testo italiano, si pone il problema di stabilire il modo con cui questo possa convivere con il resto del testo. L'unica regola sicura è la verifica dell'uso generale. Tuttavia si possono definire alcune regole di massima, per dare l'idea del problema.

- La prima cosa da fare di fronte a un termine da non tradurre è di verificare in un vocabolario di lingua italiana; se c'è, il problema potrebbe essere considerato come risolto.
- Un termine straniero può assumere il genere che avrebbe se fosse tradotto in italiano, oppure quello che suona meglio dandogli un significato italiano. In caso di dubbio è importante controllare l'uso comune (se esiste).
- I termini stranieri non tradotti sono invariabili al plurale, cioè quando sono inseriti in testi in italiano vanno scritti sempre al singolare, senza aggiungere la lettera «s» finale, anche se ci si riferisce a una quantità maggiore di uno (si pensi al termine «mouse» che al plurale inglese diventa «mice»), perché la lingua italiana non può incorporare le regole di un'altra lingua.

Quando il termine che non si traduce non è di uso comune nell'ambiente a cui si rivolge il documento, dovrebbe essere evidenziato in

corsivo tutte le volte che viene utilizzato. Per chiarire meglio il concetto, un termine tecnico può essere o meno di uso comune per il pubblico di lettori a cui si rivolge: se si tratta di un termine considerato normale per quell'ambiente, non è il caso di usare alcuna evidenziazione.

47.3.2 Anglofilia eccessiva

«

L'influenza della lingua inglese porta a deformazioni sempre più frequenti nella lingua italiana. Il problema più evidente, ma più facile da affrontare, è quello dei «falsi amici»: quei termini che, pur assomigliandosi (e pur avendo, spesso, la stessa etimologia), hanno significati diversi nelle due lingue. Gli esempi più celebri sono «factory» che diventa erroneamente «fattoria» e «cold» che si trasforma in «caldo».

Il problema meno evidente e per questo più insidioso è dato dalle altre differenze fra le due lingue: la punteggiatura, l'uso delle maiuscole e la struttura delle frasi. Trascurando queste particolarità si rischia di ottenere un testo che è formalmente in italiano, ma che non «suona» come tale.

Tabella 47.19 Traduzioni corrette dei «falsi amici».

consistent	coerente
exhaustive	esauriente
line	riga (quasi sempre)
re... (recursive)	ri... (ricorsivo)
set	insieme («set» è tennistico)
to set	impostare («settare» è di pessimo gusto)
subject	oggetto (di una lettera o di un messaggio)
to process	elaborare
to assume	supporre

proper (agg.)	giusto, corretto
proper (avv.)	vero e proprio
to support	si usi, per quanto possibile, una perifrasi
to return something	restituire qualcosa («ritornare» è intransitivo)

Quello che segue è un elenco di annotazioni riguardo all'uso dell'ortografia e della sintassi, influenzata erroneamente dalle consuetudini anglofone.

- La «e» o la «o» che introduce l'ultimo termine di un elenco non va preceduta da virgola. In inglese americano la norma è di usare la virgola (ma gli inglesi non la usano); a volte in italiano la virgola è ammissibile, ma si tratta di eccezioni.
- Se le frasi sono negative devono essere separate con «né». Per esempio, una frase del tipo «non possono essere cancellati o modificati» va espressa piuttosto come «non possono essere né cancellati, né modificati».
- I periodi italiani sono più complessi di quelli inglesi, a parità di registro, utilizzando opportunamente le congiunzioni, le subordinate, i due punti o i punti e virgola.
- I nomi dei mesi sono minuscoli.
- I numeri (interi) che esprimono una quantità piccola vanno scritti preferibilmente per esteso.
- In italiano si usa, di solito, la sequenza nome+aggettivo; il contrario, aggettivo+nome, per quanto accettabile, ha spesso un significato diverso. Per esempio, si osservi la differenza tra «pover'uomo» e «uomo povero».

- Bisogna sempre concordare il genere grammaticale: forme del tipo «la directory padre» non hanno senso.

47.3.3 Unità di misura

«

Nella documentazione a carattere scientifico diventa fondamentale la coerenza e la precisione nel modo in cui si indicano le grandezze e le unità di misura, oltre che la scelta di queste. In generale, ogni ambiente tecnico particolare tende a utilizzare le proprie grandezze e le proprie unità di misura, tralasciando gli sforzi di standardizzazione internazionale, contribuendo così a complicare inutilmente il proprio settore.

Purtroppo, l'ambito informatico costituisce l'esempio più problematico sotto questo aspetto, dal momento che l'esigenza di mantenere una compatibilità con il sistema binario ha attribuito a delle denominazioni ben precise del sistema decimale un significato differente rispetto a quello comune a tutti gli altri ambiti scientifici.

Lo standard internazionale sulle unità di misura è costituito dal SI, ovvero *Le Système international d'unités*, in italiano *Sistema internazionale di unità*. Il punto di riferimento per questo lavoro di armonizzazione è il BIPM (*Bureau international des poids et mesures*), con sede in Francia (*http://www.bipm.org/*).

47.3.3.1 Come si scrive una grandezza

«

Per esprimere una quantità riferita a una grandezza in modo grafico, occorre disporre del ***simbolo*** (la sigla) che ne esprime l'unità di misura o un multiplo opportuno di tale unità, al quale si fa precedere il numero, in cifre, di tale quantità:

n simbolo

È importante che tra il numero e la sigla ci sia uno spazio, il quale non deve poter essere interrotto in fase di impaginazione del testo. Per esempio: si può scrivere 5 kg, ma non 5kg.

47.3.3.2 Nomi e simboli

È bene chiarire il significato di alcuni termini che riguardano la misurazione di qualcosa:

Termine	Definizione
grandezza	ciò che viene misurato, come la lunghezza, la massa, il tempo;
unità di misura	il nome attribuito a ciò che si usa per misurare, come il metro, il kilogrammo, il secondo;
simbolo	il simbolo che rappresenta l'unità di misura in modo standard, come «m», «kg», «s».

Secondo il SI, il kilogrammo è l'unità di misura della massa, tenendo conto che i prefissi si utilizzano facendo riferimento al grammo. Si osservi inoltre che non si parla di «peso», perché questo termine è riferito piuttosto alla forza applicata a un oggetto.

I nomi delle unità di misura si esprimono generalmente senza iniziale maiuscola, mentre i simboli usati per rappresentarle simbolicamente vanno espressi esattamente come stabilito dagli standard, per quanto riguarda l'uso delle lettere maiuscole o minuscole.

Tabella 47.21. Esempi di grandezze e unità di misura.

Grandezza	Unità di misura	Simbolo
lunghezza	metro	m
massa	kilogrammo	kg
tempo	secondo	s
corrente elettrica	ampere	A

47.3.3.3 Prefissi moltiplicatori

«

Oltre alla definizione dei simboli che esprimono le unità di misura, si aggiungono dei simboli che rappresentano un multiplo ben preciso di tali unità. Tali simboli di moltiplicazione si pongono davanti al simbolo di unità a cui si riferiscono; per esempio, il simbolo «km» rappresenta mille unità «m», ovvero mille volte il metro.

I simboli che rappresentano tali moltiplicatori hanno anche un nome che normalmente si esprime senza iniziale maiuscola, indipendentemente dalla forma, maiuscola o minuscola, che ha il simbolo stesso.

I moltiplicatori riferiti alle unità di misura hanno un significato e un valore ben preciso. È un errore l'uso dei termini «kilo», «mega», «giga» e «tera», per rappresentare moltiplicatori pari a 2^{10}, 2^{20}, 2^{30} e 2^{40}, come si fa abitualmente per misurare grandezze riferite a bit o a byte.

Tabella 47.22. Prefissi del *Sistema internazionale di unità* (SI).

Nome	Simbolo	Valore	Note
yotta	Y	10^{24}	
zetta	Z	10^{21}	
exa	E	10^{18}	
peta	P	10^{15}	
tera	T	10^{12}	
giga	G	10^{9}	
mega	M	10^{6}	
kilo	k	10^{3}	Lettera «k» minuscola.
hecto, etto	h	10^{2}	
deca	da	10	
		1	Nessun moltiplicatore.
deci	d	10^{-1}	
centi	c	10^{-2}	
milli	m	10^{-3}	
micro	μ	10^{-6}	
nano	n	10^{-9}	
pico	p	10^{-12}	
femto	f	10^{-15}	
atto	a	10^{-18}	
zepto	z	10^{-21}	
yocto	y	10^{-24}	

47.3.3.4 Prefissi per multipli binari

Lo standard IEC 60027-2 introduce un gruppo nuovo di prefissi da utilizzare in alternativa a quelli del SI, per risolvere il problema dell'ambiguità causata dall'uso improprio dei prefissi del SI in ambito informatico. A questo proposito, una discussione particolareggiata su questo argomento si può trovare nel documento *Standardized*

Units for Use in Information Technology, di Markus Kuhn, http://www.cl.cam.ac.uk/~mgk25/information-units.txt . La tabella 47.23 riporta l'elenco di questi prefissi speciali.

Tabella 47.23. Prefissi IEC 60027-2.

Origine	Nome	Simbolo	Valore	Note
kilobinary	kibi	Ki	2^{10}	Si usa la «K» maiuscola.
megabinary	mebi	Mi	2^{20}	
gigabinary	gibi	Gi	2^{30}	
terabinary	tebi	Ti	2^{40}	
petabinary	pebi	Pi	2^{50}	
exabinary	exbi	Ei	2^{60}	
zettabinary	zebi	Zi	2^{70}	
yottabinary	yobi	Yi	2^{80}	

47.3.4 Rappresentazione di valori

«

La rappresentazione di valori numerici tende a seguire forme differenti a seconda del contesto e delle convenzioni nazionali. Nella *Guide for the Use of the International Systems of Units (SI)*, pubblicato dal NIST (*National institute of standards and technology*), si trovano alcuni criteri per risolvere il problema in modo non ambiguo, validi anche al di fuori della realtà inglese.

47.3.4.1 Valori percentuali

«

In generale, l'uso del simbolo '`%`' va inteso come una forma abbreviata per 0,01 e in questo modo va usato, senza eccedere. In particolare, il simbolo di percentuale va posto dopo un valore numerico, staccato da questo, ma non separabile in fase di composizione tipografica:

```
n %
```

Per esempio, si può scrivere ‘*x* = 0,025 = 2,5 %’, mentre non è corretta la forma ‘*x* = 0,025 = 2,5%’.

47.3.4.2 Valori numerici

Nella lingua italiana, come in molte altre, si usa la virgola come segno di separazione tra la parte intera e quella decimale, mentre nei paesi di lingua inglese, si utilizza il punto. A parte il problema di scegliere il segno opportuno in base alle proprie convenzioni nazionali, si pone piuttosto la difficoltà nel rappresentare numeri composti da una grande quantità di cifre.

La *Guide for the Use of the International Systems of Units (SI)* indica un metodo molto semplice e non equivoco: si separano le cifre a gruppi di tre, usando semplicemente uno spazio, sia prima, sia dopo il marcatore decimale, come si vede in questi esempi:

```
123 456 789
  3 456 789,012 345 6
      6 789,012 3
```

Naturalmente, lo spazio in questione non può consentire l’interruzione della riga in fase di composizione.

È ammissibile anche un’eccezione in presenza di raggruppamenti di sole quattro cifre, prima o dopo il marcatore decimale. In quel caso si può evitare la separazione:

```
1234
  23,2345
```

Un altro problema è quello della rappresentazione di valori numerici espressi con una base maggiore di 10, per i quali si utilizzano le prime 10 cifre numeriche e per il resto si usano le lettere alfabetiche. Queste lettere andrebbero utilizzate coerentemente, possibilmente in forma maiuscola.

47.3.5 Stile tipografico

La definizione dello stile tipografico è un altro punto delicato nella definizione dello stile letterario generale. Di solito, la sua preparazione, è compito del tipografo o del coordinatore di un gruppo di autori o traduttori.

Il modo migliore per stabilire e utilizzare uno stile tipografico è quello di usare un sistema SGML, attraverso cui definire un DTD che non permetta alcun dubbio nella relazione che ci deve essere tra le varie componenti di un documento. In questo modo, gli autori hanno solo il compito di qualificare correttamente le varie componenti del testo, senza pensare al risultato finale, per modificare il quale si può semmai intervenire sul sistema di conversione successivo.

Le sezioni seguenti trattano dei problemi legati alla definizione di uno stile tipografico per la redazione di documenti tecnico-informatici. L'idea è presa dalla guida di stile del gruppo di documentazione di Linux: LDP (*Linux documentation project*), ma le indicazioni si basano sulle consuetudini tipografiche italiane.

47.3.5.1 Blocchi di testo letterale

Scrivendo documenti che riguardano l'uso dell'elaboratore, si incorre frequentemente nella necessità di scrivere nomi, o intere parti di testo, che devono essere trattati in modo letterale. Possono essere no-

mi di file e directory, comandi, porzioni del contenuto di file, listati di programmi, ecc. In questi casi è sconsigliabile l'uso di un tipo di carattere proporzionale, perché si rischierebbe di perdere delle informazioni importanti. Si pensi al trattino utilizzato nelle opzioni della maggior parte dei comandi Unix: utilizzando un carattere proporzionale, attraverso un sistema di composizione come LaTeX, si otterrebbe un trattino corto, mentre due trattini posti di seguito genererebbero un trattino normale; e ancora, da tre trattini si otterrebbe un trattino largo.

47.3.5.2 Nomi di file e directory

I nomi di file, di qualunque tipo, dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio di un periodo, senza acquisire un'eventuale iniziale maiuscola.

I nomi di file eseguibili, in quanto tali, sono indicati preferibilmente senza il percorso necessario al loro avvio, ammesso che il sistema operativo non ne abbia bisogno per consentirne l'avvio.

I nomi di programmi per i sistemi Dos dovrebbero essere indicati utilizzando lettere maiuscole, senza tralasciare l'estensione.

47.3.5.3 Schermate, listati e simili

Il testo ottenuto da listati di vario tipo, come i pezzi di un programma sorgente, il risultato dell'elaborazione di un comando, o il contenuto di una schermata, possono essere rappresentati convenientemente

attraverso un ambiente di inclusione di testo letterale a spaziatura fissa.

Il problema sta nel fatto che l'ampiezza di tale testo non può superare i margini del corpo del documento, in base al tipo di impaginazione finale che si ritiene dover applicare. Infatti, tale testo non può essere continuato nella riga successiva perché ciò costituirebbe un'alterazione delle informazioni che si vogliono mostrare.

Generalmente, non è possibile superare un'ampiezza di 80 colonne, pari a quella di uno schermo a caratteri tradizionale.

47.3.5.4 Variabili di ambiente

«

I nomi di variabili di ambiente dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio o all'interno di un periodo.

A seconda del tipo di documentazione, potrebbe essere stata definita la convenzione per cui questi nomi debbano essere indicati sempre preceduti dal simbolo dollaro ('`$`').

La scelta di rappresentare le variabili utilizzando il dollaro come prefisso è motivata dalla facilità con cui queste possono essere poi identificate durante la lettura del testo. Tuttavia, una scelta di questo tipo potrebbe essere discutibile, perché il dollaro non appartiene al nome della variabile e perché potrebbe indurre il lettore a utilizzarlo sempre, anche quando negli script non si deve. Quindi, il buon senso deve guidare nella decisione finale.

47.3.5.5 Comandi e istruzioni

A volte si ha la necessità di indicare un comando, o un'istruzione, all'interno del testo normale. Per questo, è opportuno utilizzare un carattere a spaziatura fissa, come nel caso dei nomi di file e directory, però qui si pone un problema nuovo dovuto alla possibile presenza di spazi e trattini. I programmi di composizione normali tendono a interrompere le righe, quando necessario, in corrispondenza degli spazi ed eventualmente anche dei trattini. Se il comando o l'istruzione che si scrive è breve, è consigliabile l'utilizzo di spazi e trattini non interrompibili.[10]

Quando si utilizza SGML (compreso HTML), si può usare l'entità '` `' per indicare uno spazio non interrompibile, mentre se si usa solo LaTeX, è il carattere tilde ('`~`') che ha questa funzione.

Il problema del trattino non è semplice, perché non esiste un trattino generico non separabile, fine a se stesso. Di trattini ne esistono di varie misure e non sempre esistono corrispondenti per diversi tipi di programmi di composizione.

47.3.5.6 Nomi di applicativi

Quando si fa riferimento al nome di un programma si pongono due alternative: l'indicazione del file eseguibile oppure del nome attribuito dall'autore al suo applicativo.

Per comprendere la differenza, si può pensare a Apache: il servente HTTP. Non si tratta di un semplice eseguibile, ma di un applicativo composto da diverse parti, in cui l'eseguibile è '**`httpd`**'. Nello stesso modo, nel caso di Perl (il linguaggio di programmazione), si può pensare all'applicativo in generale, composto dalle librerie e tutto ciò

che serve al suo funzionamento; oppure si può voler fare riferimento solo all'eseguibile: '**`perl`**'.

I nomi di programmi applicativi dovrebbero essere indicati nello stesso modo in cui lo fa il loro autore, rispettando l'uso delle maiuscole e delle minuscole, in qualunque posizione del testo.

I nomi di questi tipi di entità non dovrebbero essere evidenziati in modo particolare.

47.3.5.7 Concetti e termini nuovi

«

I concetti e i termini che non si ritengono familiari per il lettore, dovrebbero essere evidenziati la prima volta che si presentano.

Per questo tipo di evidenziazione si utilizza un neretto oppure un corsivo. L'uso del neretto è contrario alla tradizione dei testi italiani, in cui questo viene fatto normalmente utilizzando solo il corsivo. Tuttavia, il neretto si presta meglio alla composizione in formati molto diversi; per esempio si ottiene facilmente anche su un documento da visualizzare attraverso uno schermo a caratteri.

47.3.5.8 Nomi proprietari e logotipi

«

L'indicazione di nomi che fanno riferimento a marchi di fabbrica o simili, va fatta come appare nel copyright o nella nota che fa riferimento al brevetto, rispettando l'uso delle maiuscole e dell'eventuale punteggiatura. Si dovrebbe evitare, quindi, di prendere in considerazione un eventuale logo grafico del prodotto, in quanto non è il caso di fare risaltare maggiormente i nomi di questo tipo. Ma a questa regola si può aggiungere che, nel caso il nome sia scritto utilizzando solo lettere maiuscole, può essere opportuno limitarsi a indicarlo utilizzando solo l'iniziale maiuscola, lasciando il resto in minuscolo.

All'interno del testo non è conveniente fare riferimento a detentori di copyright o di brevetti; eventualmente, di questo problema dovrebbero farsi carico delle note opportune all'inizio del documento che si scrive (ma solo se lo si ravvisa necessario per qualche ragione particolare). A tale proposito si osservi che in generale non è indispensabile fare alcun tipo di riferimento di questo genere, se lo scopo di ciò che si scrive non è quello di trattare espressamente di questo o di quel prodotto.

47.3.5.9 Titoli

«

Nei testi di lingua italiana, i titoli vanno scritti come se si trattasse di testo normale, con le particolarità seguenti:

- non viene mai posto il punto fermo finale;
- si cerca di evitare l'inserzione di altri segni di punteggiatura, a meno che ciò sia necessario per qualche motivo;
- non si usano evidenziazioni particolari di parole o nomi come invece potrebbe avvenire nel testo normale.

Un documento a carattere tecnico viene suddiviso normalmente in segmenti a più livelli, ma in generale è bene limitare la profondità di questi, per quanto possibile.

47.3.5.10 Didascalie

«

Gli elementi che non fanno parte del flusso normale di un documento, come tabelle e figure, sono accompagnate generalmente da un titolo e da una didascalia. Il titolo serve a identificarle, mentre la didascalia ne descrive il contenuto.

I titoli di tabelle, figure e oggetti simili, seguono le regole dei titoli normali, mentre il testo delle didascalie segue le regole del testo normale. Tuttavia, quando si utilizzano programmi di composizione che permettono di abbinare solo una nota descrittiva, che funga sia da titolo, sia da didascalia, occorre fare una scelta:

- quando le note sono brevi, è opportuno che si comportino come i titoli, cioè non contengano simboli di punteggiatura;
- quando sono più lunghe, si può decidere di trattarle come didascalie vere e proprie, con tutti i simboli di punteggiatura necessari per una comprensione corretta del contenuto.

Naturalmente, la scelta fatta deve valere per tutte le descrizioni che si abbinano a questi oggetti di un particolare documento: brevi o lunghe che siano.

47.3.5.11 Elenchi descrittivi

«

Gli elenchi descrittivi, come quelli che si ottengono con HTML utilizzando la struttura seguente, possono essere insidiosi, perché potrebbero tradursi in modo differente a seconda del tipo di programma di composizione utilizzato.

```
<dl>
<dt>Primo elemento</dt>
<dd>
    <p>Descrizione del primo elemento,...
    Bla bla bla...</p>
</dd>
</dl>
```

L'elemento descrittivo dell'elenco è in pratica un titolo che introduce una parte di testo generalmente rientrata. Sotto questo aspetto,

la voce descrittiva segue le regole già viste per i titoli. Tuttavia, il problema sta nel fatto che si potrebbe essere indotti a riprendere un discorso lasciato in sospeso quando veniva introdotto l'elenco, come nell'esempio seguente:

```
Bla bla bla bla…

Primo elemento

        Descrizione del primo elemento,…
        Bla bla bla…

Qui si riprende il discorso precedente all'elenco
descrittivo.
…
```

Infatti, l'utilizzo dei rientri fa percepire immediatamente la conclusione dell'elenco stesso. Quando si scrive un documento che deve poter essere convertito in molti formati differenti, che quindi potrebbe essere elaborato da programmi di composizione di vario tipo, può darsi che i rientri vengano perduti e gli elementi descrittivi dell'elenco appaiano come dei titoli veri e propri. Ma se ciò accade, quando si ricomincia «il discorso lasciato in sospeso», sembra che questo appartenga all'argomento dell'ultimo titolo apparso.

```
Bla bla bla bla…

Primo elemento

Descrizione del primo elemento,…
Bla bla bla…

Qui si riprende il discorso precedente all'elenco
descrittivo.
…
```

Pertanto, se si vogliono utilizzare strutture di questo tipo, è consigliabile che appaiano alla fine di una sezione, quando quello che viene dopo è un titolo di una sezione o di qualcosa di simile.

47.3.5.12 Richiami in nota

«

I richiami in nota (le note a piè pagina e quelle alla fine del documento) sono composti con le stesse regole del testo normale. Quando il riferimento a una nota si trova alla fine di una parola cui segue un segno di interpunzione, è opportuno collocare tale riferimento dopo il simbolo di interpunzione stesso.

47.3.5.13 Indicizzazione

«

La costruzione di un indice analitico deriva dall'inserzione di riferimenti all'interno del testo, attraverso istruzioni opportune definite dal tipo di programma usato per la composizione.

Le voci inserite in questi riferimenti, che poi vanno a formare l'indice analitico, devono essere scelte in modo uniforme, secondo alcune regole molto semplici.

- Si utilizzano le lettere minuscole, a meno che si tratti di nomi particolari che vanno sempre scritti in un modo prestabilito:
  - i nomi proprietari vanno scritti come indicato dalla casa produttrice;
  - i nomi di applicativi software vanno scritti come indicato dall'autore;
  - i nomi di file e directory vanno scritti esattamente come sono, tenendo conto che i file eseguibili vanno indicati senza percorso, mentre gli altri potrebbero contenerlo;
  - i nomi di variabili di ambiente vanno scritti esattamente come sono, eventualmente prefissati dal simbolo dollaro.
- Si utilizza solo il singolare.

I riferimenti per la generazione dell'indice analitico vanno posti preferibilmente nei luoghi opportuni, in modo da evitare inutili rimandi a pagine che non contengono ciò che si cerca. Per esempio, la parola **file** potrebbe trovarsi in quasi tutte le pagine di un testo di informatica, mentre è conveniente che l'indice analitico riporti solo le pagine in cui si parla del concetto che questa parola rappresenta.

I nomi di programmi eseguibili e di file di dati standard, come per esempio i file di configurazione, dovrebbero essere inseriti nell'indice analitico ogni volta che appaiono nel testo.

47.3.5.14 Riferimenti bibliografici e simili

Esiste una forma precisa e molto articolata per la stesura delle bibliografie, corrispondente allo standard ISO 690. A ogni modo, vale la regola generale per cui un riferimento bibliografico deve contenere tutti i dati necessari a reperire il documento a cui si fa riferimento.

In condizioni normali, le informazioni essenziali per identificare una pubblicazione sono quelle seguenti:

- l'autore o gli autori;
- il titolo completo;
- l'editore;
- la data di edizione;
- il numero ISBN (se disponibile);
- l'URI (se il documento è disponibile attraverso la rete).

Generalmente è consigliabile comporre gli elenchi bibliografici indicando le opere a partire dall'autore, mettendo il titolo in testo corsivo o inclinato, separando le varie componenti di ogni riferimento bibliografico attraverso delle virgole, come nell'esempio seguente:

> Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4

Se non si dispone di un sistema automatico per la gestione dei riferimenti bibliografici, quando si cita un documento all'interno del testo, è bene seguire alcune regole elementari.

- I riferimenti ad altri documenti, all'interno del testo normale, vanno fatti indicando il titolo completo, in corsivo o inclinato, aggiungendo il nome dell'autore o degli autori.
- Il titolo è separato con una virgola da un eventuale sottotitolo.
- I riferimenti a un testo già citato possono essere fatti utilizzando solo il titolo o solo l'autore, o attraverso altri mezzi, purché si sia certi di non creare ambiguità o disagio al lettore.

Segue un esempio molto semplice di come può essere fatto un riferimento del genere all'interno del testo normale:

> ... Questa sezione fa riferimento a concetti contenuti in *LaTeX, Guida a un sistema di editoria elettronica*, di Claudio Beccari...

47.4 Evoluzione dell'editoria elettronica

Con il termine «editoria elettronica», si vuole fare riferimento agli strumenti utilizzabili per produrre documentazione di buona qualità dal punto di vista tipografico. L'approccio di un programma per l'editoria può essere fondamentalmente di due tipi:

- a composizione visuale o WYSIWYG (*What you see is what you get*);[11]
- a composizione differita.

Nel primo caso, durante la stesura, il documento appare sullo schermo con lo stesso aspetto che avrebbe se venisse stampato in quel momento. Nel secondo, si scrive un file di testo normale con l'inserimento di comandi, come se si trattasse di un linguaggio di programmazione; quindi si passa alla composizione (una sorta di compilazione) attraverso la quale viene generato normalmente il file finale pronto per essere inviato alla stampa.

Il primo tipo di composizione è decisamente più pesante sotto l'aspetto elaborativo, prestandosi in particolare per i documenti brevi. Il secondo ha lo svantaggio di non permettere la verifica del risultato finale fino a quando non avviene la composizione, però richiede

solo l'utilizzo di un programma normalissimo per la creazione e la modifica di file di testo, mentre solo al momento della composizione c'è bisogno di un'elaborazione significativa. In questo senso è più adatto alla redazione di documenti di grandi dimensioni.

47.4.1 Evoluzione

«

L'editoria elettronica non è più solo cartacea. In particolare esistono gli ipertesti, cioè documenti elettronici la cui consultazione avviene attraverso riferimenti e non in modo puramente sequenziale.

In questo senso, se l'editoria elettronica viene vista come mezzo di documentazione generale non più orientata a un supporto particolare, non può avere immediatamente una rappresentazione finale definitiva. Per esempio, un documento in HTML non può mai essere identico a un documento stampato.

Quando si vuole produrre un documento compatibile con diversi tipi di supporti (carta, ipertesto HTML, guida interna, ecc.) non si possono avere pretese stilistiche particolari; quindi, un programma visuale diventa quasi inutile.

A fianco di questi problemi di compatibilità, si aggiungono delle esigenze nuove, come per esempio la possibilità di estrarre dal documento elettronico determinati tipi di informazioni necessarie ad alimentare una base di dati. In questo senso, le informazioni cercate, oltre che riconoscibili all'interno del formato utilizzato, devono essere coerenti e complete.

Comunque, anche nell'ambito dell'editoria cartacea tradizionale, la prima esigenza che è stata sentita è quella dell'uniformità stilistica, cosa che sarebbe bene fosse controllabile anche attraverso il sistema elettronico di composizione.

47.4.2 Codifica del testo (markup)

Il termine *markup* (o marcatura) deriva dall'ambiente tipografico dove è stato usato per definire le annotazioni fatte su una bozza, allo scopo di segnalare al compositore o al dattilografo il modo con cui alcune parti del testo andavano evidenziate o corrette. A tale proposito, esiste uno standard nella simbologia da utilizzare in questi casi, rintracciabile ancora nei libri di tipografia. Queste annotazioni simboliche possono riferirsi all'aspetto dei caratteri, all'allineamento dei paragrafi, alle spaziature e via dicendo.

Nell'editoria elettronica, il concetto alla base del termine *markup* si è esteso in modo da includere i simboli speciali, o meglio, la codifica inserita nel testo per permetterne l'elaborazione.

Volendo generalizzare, la codifica del testo è tutto ciò che ne esplicita l'interpretazione. A livello umano, la stessa punteggiatura e certe forme di spaziatura, sono la codifica che serve a chiarire il significato del testo, diventando parte essenziale di questo. Oggi non sarebbe comprensibile separare concettualmente la punteggiatura dal testo, però in passato è stato così. Basta pensare ai telegrammi, o all'apparizione di questi simboli nella storia della scrittura.

47.4.2.1 Linguaggio di markup

La tecnica di composizione del testo utilizzando l'inserimento di marcatori o di codici, richiede la definizione di una serie di convenzioni, tali da definire un ***linguaggio di markup***. Un tale linguaggio deve specificare quale tipo di marcatura è utilizzabile, quale è richiesta, in che modo si distingua dal testo e quale sia il suo significato.

I linguaggi di *markup* possono essere diversi e si distinguono due gruppi fondamentali: linguaggi procedurali e linguaggi descrittivi.

Un linguaggio di *markup* procedurale serve a definire il processo da svolgere in un punto particolare del documento. È come un linguaggio di programmazione in cui si usano chiamate di funzioni, o di procedure, per compiere le operazioni richieste. Per esempio può trattarsi di ordini riferiti alla scrittura del testo, allo spostamento, alla definizione di margini, del salto pagina e di tutto ciò che si rende necessario. In questo senso, un linguaggio di *markup* procedurale consente generalmente la definizione completa di tutto ciò che serve a stabilire l'aspetto finale del documento stampato (o visualizzato).

Un linguaggio di *markup* descrittivo, al contrario, usa la codifica dei marcatori per classificare le parti del documento, dando loro un nome. In pratica, si delimitano queste porzioni di testo e si definisce la loro appartenenza a una categoria determinata, identificata da un nome. In tal modo, questo tipo di linguaggio di *markup* non è in grado di fornire indicazioni sull'aspetto finale del documento, in quanto il suo scopo è solo quello di definire la struttura del testo. Evidentemente è compito di un'altra applicazione utilizzare le informazioni sulla struttura del testo per generare un formato finale, secondo regole e definizioni stabilite al di fuori del linguaggio descrittivo stesso.

47.4.2.2 Vantaggi di un linguaggio descrittivo

«

Un linguaggio di *markup* descrittivo, nel momento in cui non si prende carico di definire l'aspetto finale del documento, pone l'accento sul contenuto e non sull'apparenza. Questo è fondamentale quan-

do il «documento» viene inteso come informazione pura che possa materializzarsi in forme molto diverse.

L'informazione «pura», in quanto tale, richiede anche che sia espressa attraverso un formato indipendente dalle piattaforme, ma soprattutto che sia indipendente dai formati proprietari.

47.4.3 SGML

SGML è un linguaggio di *markup* descrittivo, definito dallo standard *ISO 8879: Information processing---Text and office systems---Standard Generalized Markup Language (SGML)*, 1986. L'SGML è uno standard internazionale per la definizione di metodi di rappresentazione del testo in forma elettronica in modo indipendente dall'hardware e dal sistema utilizzato.

Il linguaggio SGML utilizza il concetto di «tipo di documento» e di «definizione del tipo di documento». Per la precisione di parla di DTD, ovvero, *Document type definition*. In pratica, nell'ambito dell'SGML, è necessario che sia stato definito il modo in cui i vari elementi del testo possono essere utilizzati. Ciò che non è definito, non può essere usato, ma quello che è stato definito deve rispettare le regole stabilite.

A titolo di esempio, si può immaginare la definizione di un tipo di documento riferito alla scrittura di lettere commerciali. La lettera deve contenere degli elementi essenziali: il mittente, uno o più destinatari, la data, l'oggetto, il corpo, l'indicazione di colui che la firma e la sigla del dattilografo che la scrive materialmente. Tutti questi elementi devono essere presenti, probabilmente anche con un certo

ordine (l'indicazione di chi firma deve trovarsi in fondo e non all'inizio). Inoltre, questi elementi possono scomporsi in altri elementi più dettagliati; per esempio, l'informazione sulla persona che firma può comporsi della qualifica, il titolo personale, il nome e il cognome. Il DTD deve prendersi carico di definire tutto questo, stabilendo ciò che è valido e cosa invece non lo è.

In questo modo, poi, un documento SGML può essere analizzato da un programma speciale, l'analizzatore SGML (*SGML parser*), per la verifica del rispetto di queste regole, prima di utilizzare in qualunque modo questo documento.

L'SGML, assieme al DTD, garantendo l'uniformità dei documenti dello stesso tipo, consente di uniformare i procedimenti successivi. Per tornare all'esempio precedente, da un punto di vista di puro contenuto del testo, non dovrebbe essere importante l'ordine degli elementi che lo compongono, quando sia possibile distinguerli. Tuttavia, una lettera che inizia con la firma e finisce con l'indicazione del destinatario, non è scritta nel modo corretto; così il DTD potrebbe essere progettato in modo da imporre un certo ordine, a vantaggio delle elaborazioni successive.

47.4.4 XML, XSLT e XSL-FO

«

XML (*Extensible markup language*) è un linguaggio derivato da SGML, nato inizialmente come sottoinsieme «compatibile» con questo. A XML si affianca XSLT (*Extensible stylesheet language transformations*), come linguaggio di definizione della trasformazione di un documento XML in qualcosa di differente. XSL-FO (*Extensible stylesheet language formatting object*) è un linguaggio di definizione del formato finale del documento; spesso si usa XSLT per

ottenere un documento XSL-FO, per poi passare alla composizione finale.

47.5 Lettera, codifica e carattere da stampa

La codifica dei caratteri, intendendo ciò come il modo di rappresentare i simboli tipografici in forma elettronica, diventa un problema serio nel momento in cui si esce dallo schema abituale della scrittura con caratteri latini.

Nella storia dell'informatica è stata definita una quantità enorme di codifiche differenti, per adattare la limitatezza degli 8 bit tradizionali all'insieme di caratteri che serve in ogni circostanza particolare. Inoltre, nelle situazioni in cui 8 bit non possono bastare, sono state ideate codifiche attraverso l'abbinamento di sequenze di simboli elementari che ne rappresentano uno più complesso.

In generale, verrebbe da pensare che sarebbe stato meglio prevedere subito il problema, definendo delle unità di codifica più grandi (non più il byte di 8 bit, ma stringhe binarie più lunghe). Tuttavia, c'è da considerare che proprio la semplicità dell'alfabeto inglese (che non ha nemmeno le lettere accentate) ha permesso lo sviluppo rapido di tecnologia relativamente «semplice», che altrimenti sarebbe stata materialmente irraggiungibile.

Il byte stesso (quello da 8 bit) è stata una grande conquista. Ancora oggi ci sono sistemi di comunicazione che riconoscono unità di codifica a soli 7 bit, dove in pratica si può usare solo l'ASCII; prima ancora sono state utilizzate anche unità di codifica a soli 6 bit.

Per comprendere il problema della codifica, è necessario considerare i problemi che riguardano la definizione dei caratteri da stampa.

La prima fase è la definizione di un repertorio astratto, all'interno del quale si elencano, senza un ordine preciso, le lettere e gli altri segni necessari per un certo scopo. L'insieme di questi simboli è astratto, nel senso che non è ancora stabilito l'aspetto finale, compito riservato a una fase successiva.[12]

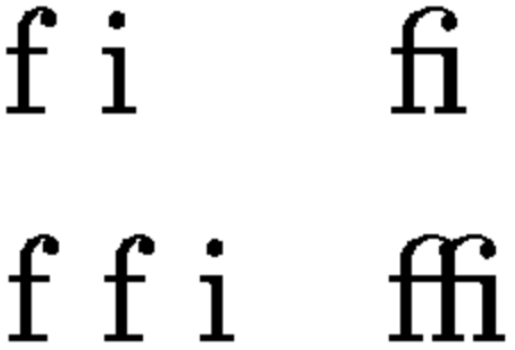
Il simbolo di un repertorio astratto è qualcosa di diverso dal simbolo che compone un carattere da stampa, dal momento che il secondo rappresenta il modo preciso in cui il simbolo astratto viene reso tipograficamente. Per comprendere il concetto, si pensi alla lettera «a» e all'aspetto che questo simbolo astratto può avere utilizzando stili, serie e corpi differenti. Evidentemente, si tratta sempre della stessa lettera, ma resa in modo diverso.

Figura 47.29. La lettera «a» minuscola resa tipograficamente in modo differente.

Alcuni gruppi di simboli astratti tendono a essere rappresentati tipograficamente in un simbolo solo, in un legato, ovvero attraverso l'avvicinamento e la sovrapposizione parziale. Il caso tipico è rappresentato dalla sequenza di lettere «fi» e «ffi», come si vede nella figura 47.30. In certi casi, la sequenza di lettere che si avvicina rappresenta una parola intera, generando così un «logotipo»; spesso, la loro importanza storica ha fatto sì che questi siano diventati dei simboli astratti autonomi. Per esempio, la parola latina «et» è diventata la e-commerciale odierna, «&»; la parola latina «ad» è diventata la chiocciola odierna, «@» (nella lingua italiana si è perso l'uso di questo legato, riacquisendolo solo attraverso la lingua inglese, pertanto

lo si conosce solitamente solo nella sua definizione inglese: *at*).

Figura 47.30. Il legato «fi» e «ffi».



La composizione tipografica elettronica, può avvenire attraverso la sovrapposizione di simboli elementari differenti, senza la necessità di legarli assieme. Nelle lingue di origine latina, il caso più comune di questa possibilità si ha con gli accenti, i quali potrebbero essere simboli tipografici separati da sovrapporre alle lettere a cui sono destinati. Nello stesso modo, il repertorio di simboli astratti potrebbe essere realizzato con questo criterio; per esempio, per fare riferimento a un simbolo complesso potrebbe essere necessario indicare una sequenza di simboli astratti elementari.

Alcune lingue hanno dei simboli che nella composizione tipografica devono cambiare forma a seconda del contesto. Per comprendere il concetto, si può pensare a una scrittura manuale, in cui le lettere cambiano leggermente forma a seconda di ciò che appare prima e dopo; chi conosce una scrittura stenografica manuale, può intendere ancora meglio il problema. Ad aggravare ancora di più la questione, l'adattamento contestuale di un simbolo potrebbe dipendere da una scelta stilistica, in parte arbitraria.

A volte, la larghezza di un testo deve essere adattata per esigenze estetiche, come avviene nel caso dell'allineamento simultaneo a si-

nistra e a destra. Nelle lingue di origine latina si ottiene questo attraverso l'allargamento degli spazi tra le parole e tra le lettere all'interno delle parole; tuttavia, alcune lingue richiedono degli adattamenti differenti, per esempio attraverso l'introduzione di altri simboli appropriati.

Da quello che è stato scritto si intende che la composizione tipografica elettronica si può considerare come l'ultima fase di un processo composto da tre livelli: definizione di un repertorio astratto di simboli; definizione di una codifica; composizione tipografica a partire dalla codifica.

repertorio astratto ⟶ codifica ⟶ composizione

La codifica non può corrispondere esattamente al repertorio astratto ideale: deve fare delle scelte. In generale, il repertorio simbolico preso in considerazione dalla codifica è identificabile come un insieme di ***punti di codifica*** (*code point*, secondo la documentazione di Unicode).

I problemi legati alla composizione tipografica che sono stati descritti, sono solo alcuni di quelli che si incontrano. A seconda dei casi, implicano un approccio differente per ciò che riguarda la codifica e la composizione. In breve:

1. lo stile tipografico è qualcosa che normalmente è gestito dal sistema di composizione, senza richiedere la definizione di punti di codifica differenti;
2. il legato può essere un problema risolto a livello di composizione finale, oppure può richiedere la definizione di punti di codi-

fica aggiuntivi, quando si tratta di legati molto importanti o di logotipi;

3. l'adattamento contestuale richiede spesso la definizione di tanti punti di codifica quante sono le varianti contestuali del simbolo astratto, specialmente se esiste un margine di scelta da parte dell'autore;
4. l'adattamento della larghezza del testo dovrebbe essere compito del sistema di composizione, anche quando questo implica l'inserzione di simboli speciali.

47.6 Ambiguità nel concetto di «carattere»

In informatica, il termine «carattere» ha acquisito un significato ambiguo che dipende dal contesto. Per esempio, può riferirsi a un simbolo del repertorio astratto, a un punto di codifica, all'unità di memorizzazione (unità di codifica, o *code unit*), o al segno che viene ottenuto alla fine del processo di composizione. Per fare un esempio in merito all'unità di codifica, basta pensare al byte che spesso viene confuso con il carattere, mentre ormai è da intendersi come un'unità di memorizzazione troppo piccola per questo scopo nel sistema globale.

Di certo non si può pretendere che si smetta di usare il termine carattere per passare invece a una terminologia più precisa. Tuttavia è importante rendersi conto della vastità della cosa e dei problemi che ci stanno sotto.

Il modello di Unicode suddivide il problema della codifica in cinque livelli:

1. ACR (*Abstract character repertoire*)

definizione di un repertorio astratto di simboli;

2. CCS (*Coded character set*)

   definizione di una mappa in cui si abbina un numero intero, non negativo, a ogni simbolo del repertorio astratto che si intende gestire;

3. CEF (*Character encoding form*)

   definizione di una mappa in cui si abbinano i numeri ottenuti dal livello precedente a un insieme di sequenze dell'unità di codifica prescelta;

4. CES (*Character encoding scheme*)

   definizione di una mappa di trasformazione delle unità di codifica in una sequenza seriale di byte;

5. TES (*Transfer encoding syntax*)

   definizione di un metodo reversibile di trasformazione dei dati codificati in base alle limitazioni del mezzo trasmissivo.

Da un punto di vista leggermente differente, si potrebbe scomporre il problema in strati, per distinguere le fasi che vanno dalla scrittura alla trasmissione del testo e dalla ricezione del testo alla lettura. La scrittura potrebbe essere descritta con l'elenco di operazioni seguenti:

1. selezioni dei simboli e digitazione attraverso la tastiera (o un altro mezzo);
2. codifica, attraverso cui si trasforma il simbolo in un numero intero non negativo;
3. trasformazione in unità di codifica, in base alla forma prescelta;

4. adattamento in sequenze di byte;
5. adattamento prima del trasferimento dei dati.

Il processo di lettura dei dati, a partire dalla ricezione, è opposto:

1. interpretazione dei dati ricevuti e ricostruzione delle sequenze di byte di partenza;
2. trasformazione delle sequenze di byte in sequenze di unità di codifica;
3. trasformazione dalle sequenze di unità di codifica in numeri interi non negativi;
4. decodifica dei numeri interi non negativi;
5. composizione tipografica (su schermo o su carta).

Prima di questa sezione è già stato affrontato il problema dell'abbinamento tra il repertorio astratto di simboli e la codifica, senza precisare in che modo sia organizzata questa ultima. Nelle sezioni seguenti si accenna alle problematiche successive.

47.6.1 CCS: insieme di caratteri codificato

L'insieme di caratteri codificato è in pratica il repertorio simbolico disponibile effettivamente, ottenuto dopo la definizione di un repertorio astratto e dopo lo studio dei problemi legati alla cultura e alle consuetudini del linguaggio per il quale è stato realizzato. Questo insieme di caratteri abbina un numero intero a ogni simbolo, senza bisogno che ci sia continuità nella sequenza di tale valore (l'unica limitazione è quella per cui deve trattarsi di un valore non negativo).

Il numero che rappresenta il simbolo di un insieme di caratteri codificato, è il punto di codifica.

Nella documentazione tecnica si fa spesso riferimento al concetto di «insieme di caratteri», ovvero *character set*, per intendere quello che qui si indica come «insieme di caratteri codificato», ovvero CCS, *Coded character set*.

Alcuni esempi tradizionali di insiemi di caratteri codificati sono:

ASCII (ISO 646)	127 punti di codifica;
ISO 8859-1	i primi 127 punti di codifica sono uguali all'ASCII;
ISO 8859-2	i primi 127 punti di codifica sono uguali all'ASCII, mentre nella parte restante il repertorio dei simboli è diverso dall'ISO 8859-1.
Unicode (ISO 10646)	i primi 255 punti di codifica sono uguali a ISO 8859-1.

Alcuni insiemi di caratteri codificati prevedono l'abbinamento con una descrizione (in inglese), allo scopo di facilitarne l'identificazione. Si utilizza questa descrizione per evitare ambiguità nell'identificazione del simbolo, quando questo potrebbe essere confuso con un altro, o più semplicemente quando potrebbe essere male interpretato.

Per rappresentare un punto di codifica, basta indicare il suo numero intero (qualunque sia la sua base di numerazione). Di solito, per evitare ambiguità, quando si tratta di Unicode o di ISO 10646, si fa uso normalmente della forma '`U+`***nnnn***', oppure '`U-`***nnnnnnnn***', dove ***n*** è una cifra esadecimale. Evidentemente, la seconda forma è utile

per individuare punti di codifica più grandi. Per la precisione, questa notazione è la rappresentazione delle codifiche UCS-2 e UCS-4 a cui non si intende fare riferimento direttamente. In generale, non c'è alcun bisogno di rappresentare un punto di codifica in questo modo; tuttavia, si tratta di una simbologia immediata che dovrebbe semplificare la lettura e la comprensione del testo.

Tuttavia, in questo capitolo, per maggiore chiarezza, si preferisce l'uso di una forma differente, mediata dall'XML, '**#x*n***', dove ***n*** rappresenta una o più cifre esadecimali in base alla necessità.

In questa fase della scomposizione del problema della codifica, il «carattere» è il numero intero che rappresenta il punto di codifica. Attraverso un linguaggio di programmazione che sia adeguato al problema della codifica universale, il tipo di dati «carattere» deve corrispondere a un intero senza segno per il quale non ci si pone il problema del limite (anche se in questo momento dovrebbe essere almeno un intero a 32 bit); di conseguenza, il tipo stringa dovrebbe essere un array del tipo carattere.[13]

47.6.2 CEF: forma codificata del carattere

«

La forma codificata del carattere è il risultato di una trasformazione dal numero intero non negativo che costituisce il livello precedente, in una sequenza di unità di codifica. L'unità di codifica è un raggruppamento di bit di una lunghezza opportuna.

La sequenza di unità di codifica non è composta necessariamente dalla stessa quantità di queste unità per tutti gli elementi dell'insieme di caratteri. A tale proposito, si distingue tra forme codificate del carattere a lunghezza fissa e a lunghezza variabile.

L'esempio più semplice di forma codificata del carattere a lunghezza fissa è dato dall'ASCII tradizionale: l'insieme di caratteri codificato è costituito da 128 punti di codifica, rappresentati da tutti gli interi che vanno da 0 a 127. L'unità di codifica utilizzata in questa situazione è un gruppo singolo di 7 bit con i quali si rappresenta lo stesso numero intero.

Il caso più comune di forma codificata del carattere a lunghezza variabile è dato dall'UTF-8, che utilizza un'unità di codifica di un ottetto (un byte), in cui i punti di codifica con valori tra 0 e 127 (da 00_{16} a $7F_{16}$) utilizzano una sola unità di codifica, mentre tutti gli altri ne utilizzano più di una.

In fase di interpretazione delle sequenze di unità di codifica si possono presentare i casi seguenti:

1. la sequenza potrebbe non essere valida, perché incompleta, o perché esclusa esplicitamente;
2. la sequenza potrebbe fare riferimento a un punto di codifica possibile ma non ancora assegnato a un simbolo;
3. la sequenza potrebbe corrispondere a un punto di codifica assegnato a un simbolo stabilito, oppure lasciato all'attribuzione libera senza un vincolo preciso.

Il problema delle sequenze incomplete si intende nel momento in cui si accetta il fatto che una forma di codifica possa prevedere una

lunghezza variabile delle sequenze di unità di codifica. Il caso dei punti di codifica lasciati al libero arbitrio degli utilizzatori, è una particolarità della codifica universale (Unicode e ISO 10646); se ne può comprendere la necessità di fronte a un sistema di codifica che vuole essere completo, ma che in pratica è appena all'inizio della sua opera di catalogazione.

A questo livello della scomposizione del problema, il «carattere» è ciò che idealmente è scritto in un «file di testo» (non più solo un «file ASCII»). Anche se è stato stabilito in che modo è organizzato l'insieme di caratteri codificato, la sua rappresentazione binaria «ideale» nel file di testo dipende dalla forma prescelta. Qui si parla di rappresentazione ideale, perché la rappresentazione reale dipende dal livello successivo, in cui tutto viene tradotto a livello di byte.

47.6.3 CES: schema di codifica del carattere

Lo schema di codifica del carattere è un sistema di trasformazione attraverso il quale, le unità di codifica vengono rese in sequenze di byte messe in serie.

Per tornare all'esempio dell'ASCII, l'unità di codifica è di 7 bit, ma il «carattere» ASCII si gestisce in pratica all'interno di un byte, dove il bit più significativo viene lasciato azzerato.

In generale, il byte è un'unità di memorizzazione standard in tutte le architetture dei sistemi elaborativi e in tutti i sistemi di trasmissione dati. Questo spiega la necessità di trasferire tutto a livello di byte o di multipli di questa unità.

Dovendo utilizzare più byte per rappresentare un oggetto unico, si pone il problema dello scambio tra coppie di byte che avviene in alcune architetture. Come è noto, si distingue tra *big-endian*, in cui

il primo byte è quello più significativo, e *little-endian*, in cui il primo byte è quello meno significativo. Pertanto, in questa situazione, si impone la necessità di specificare l'ordine dei byte.

47.6.4 TES: sintassi di codifica per il trasferimento

«

La sintassi di codifica per il trasferimento è un metodo di trasformazione reversibile di una codifica, che si deve attuare a causa di qualche tipo di esigenza. Per esempio:

- la necessità di evitare l'utilizzo di alcuni valori nei byte che potrebbero confondere un sistema di comunicazione o di memorizzazione;
- la necessità di ridurre la dimensione dei dati utilizzando algoritmi di compressione.

Mentre il secondo caso dovrebbe essere chiaro, per comprendere il primo basta pensare alle limitazioni che ha storicamente il protocollo SMTP (posta elettronica), per cui è necessario evitare di trasmettere byte in cui il primo bit sia diverso da zero.

47.7 Codifica in pratica: da Unicode a ASCII

«

Il lavoro per la realizzazione dell'insieme di caratteri universale non può partire da zero, per l'esigenza di mantenere qualche forma di compatibilità con il passato (diversamente, non verrebbe nemmeno preso in considerazione). Pertanto, le incongruenze che si possono rilevare sono dovute principalmente a questo motivo: la necessità di riutilizzare gli insiemi di caratteri codificati più importanti già esistenti.

Unicode e ISO 10646 sono due standard compatibili reciprocamente che definiscono un insieme di caratteri codificato particolarmente grande, da trasformare poi nella forma codificata del carattere prescelta per la sua rappresentazione pratica in unità di codifica. Pertanto, quando di parla di Unicode, o di ISO 10646, senza specificare altro, si pensa generalmente ai punti di codifica e non alla rappresentazione finale.[14]

I primi punti di codifica di questi standard corrispondono esattamente all'ISO 8859-1. Per esempio: #x20 è lo spazio normale; #xA0 è lo spazio non interrompibile; #xAB sono le virgolette angolari aperte; #xBB sono le virgolette angolari chiuse.

Attualmente, l'insieme di caratteri universale utilizza principalmente tre forme codificate del carattere UTF (*Unicode transformation format*): UTF-8, UTF-16 e UTF-32. Ogni forma codificata del carattere del tipo UTF-***n*** rappresenta un punto di codifica come una sequenza di una o più unità di codifica (che a sua volta occupa ***n*** bit), ottenuta attraverso una trasformazione reversibile del valore.

Con questo sistema, i punti di codifica che possono essere rappresentati vanno teoricamente da #x0 a #x7FFFFFFF (in particolare, secondo Unicode si arriva solo fino a #x10FFFF), salvo alcuni valori che sono stati esclusi espressamente. I punti di codifica esclusi più importanti sono #xFFFE e #xFFFF.

Le forme codificate del carattere che utilizzano le unità di codifica più piccole, richiedono l'uso di sequenze multiple di tali unità con maggiore frequenza. Per esempio, si può osservare il caso di UTF-8, in cui l'unità di codifica è il byte (un ottetto): mano a mano che il valore del punto di codifica cresce, è necessario utilizzare più

unità di codifica per la sua rappresentazione.

È necessario sottolineare il fatto che i valori che compongono l'insieme dei punti di codifica, non vengono trasferiti tali e quali nella forma codificata, dal momento che ci possono essere delle limitazioni nella rappresentazione.

Allo stato attuale dello sviluppo dell'insieme di caratteri universale, le varie forme codificate del carattere possono utilizzare gli spazi seguenti:

UTF-8	da uno a sei unità di codifica da 8 bit (mentre secondo Unicode, che è più restrittivo dello standard ISO 10646, si hanno al massimo quattro unità);
UTF-16	da uno a due unità di codifica da 16 bit;
UTF-32	attualmente si prevede una sola unità di codifica da 32 bit.

47.7.1 UTF-8

«

UTF sta per *Unicode transformation format* e significa implicitamente che si tratta di una mappa di trasformazione da punti di codifica Unicode a unità di codifica (è già stato descritto il fatto che il numero che segue la sigla UTF-***n*** indica la dimensione in bit dell'unità di codifica).

In particolare, vale la pena di osservare un po' meglio UTF-8, essendo questo il cardine della transizione verso l'insieme di caratteri universale nei sistemi operativi in cui non è conveniente l'utilizzo di unità di codifica più grandi. In effetti, UTF-8 è un sistema molto complesso per rappresentare simboli di qualunque lingua diversa

dall'inglese, perché richiede spesso l'utilizzo di più unità per un solo simbolo.

Le caratteristiche di UTF-8 sono le seguenti:

- i punti di codifica da #x0 a #x7F, corrispondenti in pratica all'ASCII, sono tradotti semplicemente in byte da 00_{16} a $7F_{16}$, esattamente come si fa già con l'ASCII stesso;
- i punti di codifica che vanno da #x80 in su, vengono tradotti in sequenze multiple di byte, ognuno dei quali ha il bit più significativo a uno, così da evitare che i byte da 00_{16} a $7F_{16}$ possano apparire all'interno delle sequenze multiple;
- il primo byte di una sequenza multipla che rappresenta un punto di codifica che vada da #x80 in su, contiene sempre valori nell'intervallo da $C0_{16}$ a FD_{16} e serve a indicare quanti byte vengono utilizzati per rappresentare il carattere;
- i byte di una sequenza multipla che sono successivi al primo contengono valori che vanno da 80_{16} a BF_{16};
- si possono definire sequenze di byte in numero massimo di sei;
- i valori FE_{16} e FF_{16} non sono mai usati.

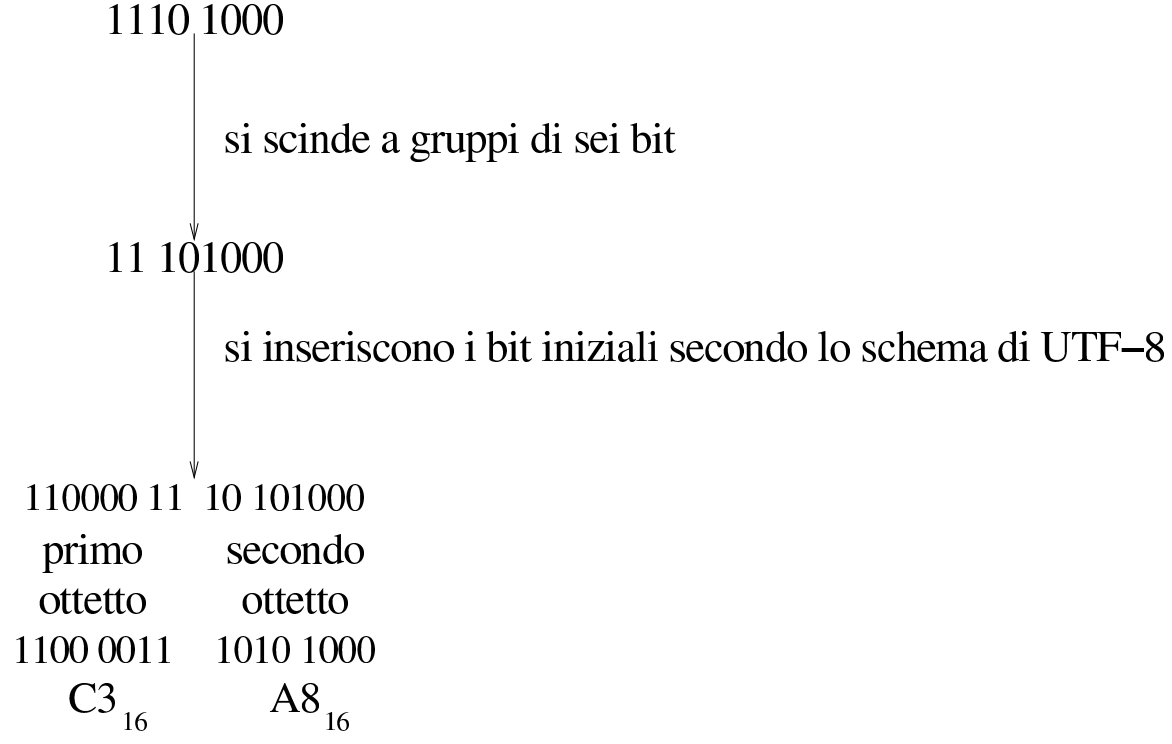
La tabella 47.34 dovrebbe chiarire meglio il concetto, abbinando i valori dei punti di codifica Unicode alle sequenze di byte con cui possono essere rappresentati. Si osservi che la lettera ***x*** serve a indicare un bit variabile.

Tabella 47.34. Sequenze multi-byte teoriche nell'UTF-8.

da	a	sequenze di ottetti
#x0	#x7F	0*xxxxxxx*
#x80	#x7FF	110*xxxxx* 10*xxxxxx*
#x800	#xFFFF	1110*xxxx* 10*xxxxxx* 10*xxxxxx*
#x10000	#x1FFFFF	11110*xxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
#x200000	#x3FFFFFF	111110*xx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
#x4000000	#x7FFFFFFF	1111110*x* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*

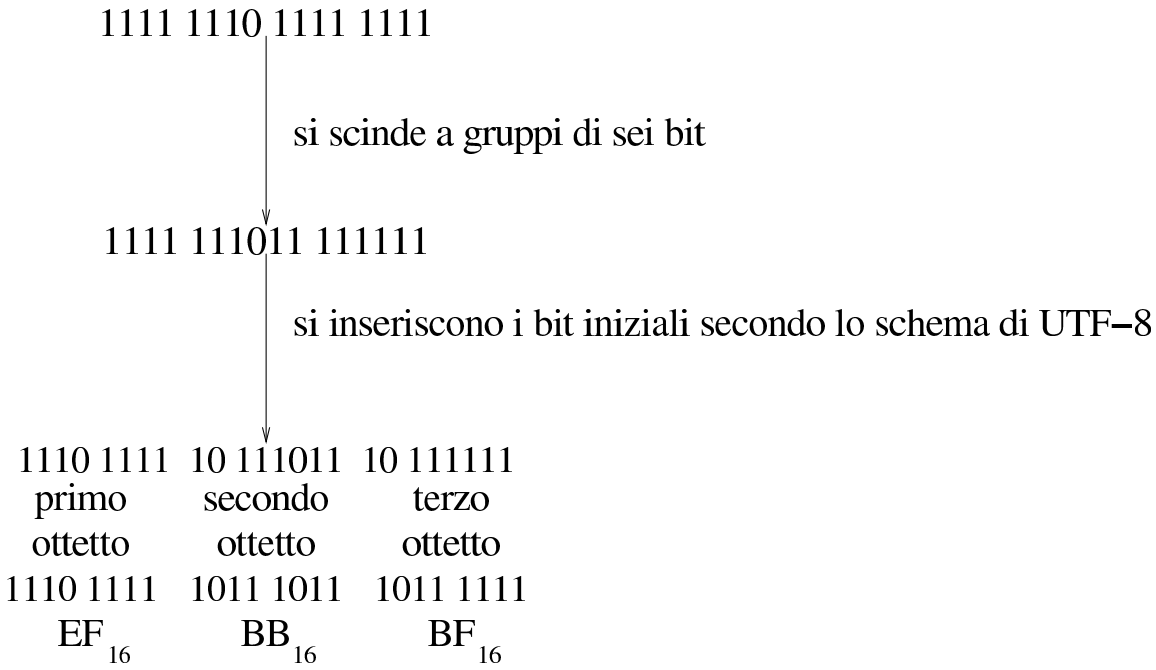
Un esempio dovrebbe chiarire ancora meglio il meccanismo. La lettera accentata «è» si rappresenta attraverso il punto di codifica #xE8, il quale in pratica si può rendere in binario come $1110\ 1000_2$, si traduce in UTF-8 come si vede nella figura 47.35.

Figura 47.35. #xE8 in UTF-8.



Un altro esempio interessante è il punto di codifica #xFEFF ($1111\ 1110\ 1111\ 1111_2$); lo si vede nella figura 47.36.

Figura 47.36. #xFEFF in UTF-8.



Da questo si dovrebbe intendere il passaggio a un numero superiore di byte.

In base al modello di UTF-8, si potrebbero realizzare anche sequenze più lunghe del necessario per rappresentare un punto di codifica. Evidentemente, è compito del software che le genera evitare di sprecare dello spazio inutilmente.

47.7.2 Schema di codifica e firma di riconoscimento

Di fronte a diverse forme codificate del carattere UTF, c'è la necessità di poterle identificare facilmente. Per questo si utilizza una sorta di firma iniziale, costituita in pratica dal punto di codifica #xFEFF, il quale, nella trasformazione in base allo schema di codifica del carattere, permette anche di controllare se l'ordine dei byte è normale o è stato invertito.

Il punto di codifica #xFEFF viene anche identificato con il nome ZWNBSP, ovvero *Zero width no-break space*; tuttavia, anche se si

intende che si tratta di qualcosa di «innocuo» (uno spazio non interrompibile di ampiezza nulla), se è stato inserito come firma iniziale, non va inteso come parte del testo. Ciò significa che i programmi per la gestione di file di testo devono tenere conto che la firma iniziale va tolta prima di fare qualunque elaborazione (si pensi al concatenamento con un comando '**`cat`**' o simile).

Gli schemi di codifica del carattere riferiti alle forme codificate UTF, si possono precisare aggiungendo delle sigle alla fine del nome UTF-***n***. La tabella 47.37 mostra gli schemi di codifica UTF-***n*** $*$, assieme alla firma iniziale (quando questa è prevista).

Tabella 47.37. Schemi di codifica UTF-***n***.

Schema	Firma iniziale	Note
UTF-8	$EFBBBF_{16}$	In condizioni normali è prevista la firma iniziale.
UTF-8N		Si indica esplicitamente l'assenza della firma.
UTF-16		UTF-16 *big-endian* in modo predefinito.
UTF-16	$FEFF_{16}$	UTF-16 *big-endian*.
UTF-16	$FFFE_{16}$	UTF-16 *little-endian*.
UTF-16BE		UTF-16 *big-endian* senza firma.
UTF-16LE		UTF-16 *little-endian* senza firma.
UTF-32		UTF-32 *big-endian* in modo predefinito.
UTF-32	$0000FEFF_{16}$	UTF-32 *big-endian*.
UTF-32	$FFFE0000_{16}$	UTF-32 *little-endian*.
UTF-32BE		UTF-32 *big-endian* senza firma.
UTF-32LE		UTF-32 *little-endian* senza firma.

47.7.3 Tipi di dati nuovi

Si è già accennato al modo in cui un linguaggio di programmazione può gestire i punti di codifica di questo tipo. Tuttavia, non si può dimenticare il passato; così, in tutte le situazioni in cui il «carattere» è implicitamente un intero senza segno a 8 bit, è necessario usare un'altra definizione per i punti di codifica: il ***carattere esteso***, ovvero *wide char*. Nello stesso modo, dovendo parlare di stringhe, se c'è bisogno di chiarire che si tratta di una stringa secondo Unicode o ISO 10646, si parla di ***stringa estesa***, ovvero di *wide string*.

47.7.4 Apparenza e realtà

La disponibilità di un sistema di codifica che faccia riferimento a un repertorio simbolico molto ampio, risolve tanti problemi del passato in cui era necessario risparmiare. Per esempio, nell'ASCII tradizionale, il trattino è unico (non si distingue la sua lunghezza) ed è anche un segno «meno». Disponendo di un repertorio molto grande, diventa importante utilizzare il simbolo giusto in base al contesto. Per esempio, la lettera latina «A» maiuscola, è diversa dalla lettera greca alfa maiuscola, anche se i due simboli possono avere lo stesso aspetto.

La descrizione che viene abbinata ai punti di codifica serve proprio per questo, in modo da evitare confusione.

Per fare un esempio più convincente, si pensi alla lettera «ß» nell'insieme ISO 8859-1. Il nome abbinato a questa lettera è «LATIN SMALL LETTER SHARP S»; come si legge non si tratta della lettera greca beta, ma di qualcosa di diverso. Per la precisione è un legato che si usa nella lingua tedesca; in mancanza del segno tipografico

può essere reso come «ss» (infatti si tratta di una lettera minuscola). Utilizzare questo simbolo al posto della lettera beta sarebbe un errore; infatti, un sistema di composizione o di lettura, potrebbe anche decidere di convertire il segno nella forma semplificata che è appena stata mostrata.

47.7.5 ASCII (ISO 646)

«

L'ASCII è una codifica molto semplice, in cui ogni punto di codifica corrisponde direttamente a un gruppo di 7 bit, inteso come un intero senza segno, senza bisogno di trasformazioni. Sulla base di questa codifica si sono sviluppate molte varianti, soprattutto a 8 bit. Tuttavia, oggi, quando si parla di ASCII si tende a fare riferimento prevalentemente allo standard originale, in cui si utilizzavano valori compresi tra 0 e 127, per rappresentare i quali bastano solo 7 bit. Eventualmente, volendo essere precisi, per fare riferimento all'ASCII tradizionale si può utilizzare la denominazione «US-ASCII».

L'ASCII non si occupa solo di definire la codifica dei segni tipografici, ma include anche dei codici di controllo, ai quali abbina un nome, ma senza potervi attribuire un significato univoco valido in tutti i contesti. Si tratta dei punti di codifica da 0 a 31 e del 127 in decimale (il punto di codifica 32 rappresenta lo spazio normale).

La tabella 47.38 mostra nel dettaglio la codifica ASCII.

Tabella 47.38. US-ASCII (ISO 646). L'ultima colonna si riferisce alla rappresentazione corrispondente nel linguaggio C.

Binario	Esadecimale	Ottale	Decimale	Carattere	<^*x*>	Linguaggio C
00000000_2	00_{16}	000_8	000_{10}	*<NUL>*		\0
00000001_2	01_{16}	001_8	001_{10}	*<SOH>*	*<^a>*	
00000010_2	02_{16}	002_8	002_{10}	*<STX>*	*<^b>*	
00000011_2	03_{16}	003_8	003_{10}	*<ETX>*	*<^c>*	
00000100_2	04_{16}	004_8	004_{10}	*<EOT>*	*<^d>*	
00000101_2	05_{16}	005_8	005_{10}	*<ENQ>*	*<^e>*	
00000110_2	06_{16}	006_8	006_{10}	*<ACK>*	*<^f>*	
00000111_2	07_{16}	007_8	007_{10}	*<BEL>*	*<^g>*	\a
00001000_2	08_{16}	010_8	008_{10}	*<BS>*	*<^h>*	\b
00001001_2	09_{16}	011_8	009_{10}	*<HT>*	*<^i>*	\t
00001010_2	$0A_{16}$	012_8	010_{10}	*<LF>*	*<^j>*	\n
00001011_2	$0B_{16}$	013_8	011_{10}	*<VT>*	*<^k>*	\v
00001100_2	$0C_{16}$	014_8	012_{10}	*<FF>*	*<^l>*	\f
00001101_2	$0D_{16}$	015_8	013_{10}	*<CR>*	*<^m>*	\r
00001110_2	$0E_{16}$	016_8	014_{10}	*<SO>*	*<^n>*	
00001111_2	$0F_{16}$	017_8	015_{10}	*<SI>*	*<^o>*	
00010000_2	10_{16}	020_8	016_{10}	*<DLE>*	*<^p>*	
00010001_2	11_{16}	021_8	017_{10}	*<DC1>*	*<^q>*	
00010010_2	12_{16}	022_8	018_{10}	*<DC2>*	*<^r>*	
00010011_2	13_{16}	023_8	019_{10}	*<DC3>*	*<^s>*	
00010100_2	14_{16}	024_8	020_{10}	*<DC4>*	*<^t>*	
00010101_2	15_{16}	025_8	021_{10}	*<NAK>*	*<^u>*	

Binario	Esadecimale	Ottale	Decimale	Carattere	<^*x*>	Linguaggio C
00010110_2	16_{16}	026_8	022_{10}	*<SYN>*	*<^v>*	
00010111_2	17_{16}	027_8	023_{10}	*<ETB>*	*<^w>*	
00011000_2	18_{16}	030_8	024_{10}	*<CAN>*	*<^x>*	
00011001_2	19_{16}	031_8	025_{10}	*<EM>*	*<^y>*	
00011010_2	$1A_{16}$	032_8	026_{10}	*<SUB>*	*<^z>*	
00011011_2	$1B_{16}$	033_8	027_{10}	*<ESC>*	*<^[>*	
00011100_2	$1C_{16}$	034_8	028_{10}	*<FS>*	*<^\>*	
00011101_2	$1D_{16}$	035_8	029_{10}	*<GS>*	*<^]>*	
00011110_2	$1E_{16}$	036_8	030_{10}	*<RS>*	*<^^>*	
00011111_2	$1F_{16}$	037_8	031_{10}	*<US>*	*<^_>*	
00100000_2	20_{16}	040_8	032_{10}	*<SP>*		
00100001_2	21_{16}	041_8	033_{10}	!		
00100010_2	22_{16}	042_8	034_{10}	"		
00100011_2	23_{16}	043_8	035_{10}	#		
00100100_2	24_{16}	044_8	036_{10}	$		
00100101_2	25_{16}	045_8	037_{10}	*%*		
00100110_2	26_{16}	046_8	038_{10}	&		
00100111_2	27_{16}	047_8	039_{10}	’		
00101000_2	28_{16}	050_8	040_{10}	(		
00101001_2	29_{16}	051_8	041_{10}	)		
00101010_2	$2A_{16}$	052_8	042_{10}	*		
00101011_2	$2B_{16}$	053_8	043_{10}	+		

Binario	Esadecimale	Ottale	Decimale	Carattere	<^x>	Linguaggio C
$0010110{0}_2$	$2C_{16}$	054_8	044_{10}	,		
$0010110{1}_2$	$2D_{16}$	055_8	045_{10}	-		
$0010111{0}_2$	$2E_{16}$	056_8	046_{10}	.		
$0010111{1}_2$	$2F_{16}$	057_8	047_{10}	/		
$0011000{0}_2$	30_{16}	060_8	048_{10}	0		
$0011000{1}_2$	31_{16}	061_8	049_{10}	1		
$0011001{0}_2$	32_{16}	062_8	050_{10}	2		
$0011001{1}_2$	33_{16}	063_8	051_{10}	3		
$0011010{0}_2$	34_{16}	064_8	052_{10}	4		
$0011010{1}_2$	35_{16}	065_8	053_{10}	5		
$0011011{0}_2$	36_{16}	066_8	054_{10}	6		
$0011011{1}_2$	37_{16}	067_8	055_{10}	7		
$0011100{0}_2$	38_{16}	070_8	056_{10}	8		
$0011100{1}_2$	39_{16}	071_8	057_{10}	9		
$0011101{0}_2$	$3A_{16}$	072_8	058_{10}	:		
$0011101{1}_2$	$3B_{16}$	073_8	059_{10}	;		
$0011110{0}_2$	$3C_{16}$	074_8	060_{10}	<		
$0011110{1}_2$	$3D_{16}$	075_8	061_{10}	=		
$0011111{0}_2$	$3E_{16}$	076_8	062_{10}	>		
$0011111{1}_2$	$3F_{16}$	077_8	063_{10}	?		
$0100000{0}_2$	40_{16}	100_8	064_{10}	@		
$0100000{1}_2$	41_{16}	101_8	065_{10}	A		

Binario	Esadecimale	Ottale	Decimale	Carattere	<^x>	Linguaggio C
$0100001{0}_2$	42_{16}	102_8	066_{10}	B		
$0100001{1}_2$	43_{16}	103_8	067_{10}	C		
$0100010{0}_2$	44_{16}	104_8	068_{10}	D		
$0100010{1}_2$	45_{16}	105_8	069_{10}	E		
$0100011{0}_2$	46_{16}	106_8	070_{10}	F		
$0100011{1}_2$	47_{16}	107_8	071_{10}	G		
$0100100{0}_2$	48_{16}	110_8	072_{10}	H		
$0100100{1}_2$	49_{16}	111_8	073_{10}	I		
$0100101{0}_2$	$4A_{16}$	112_8	074_{10}	J		
$0100101{1}_2$	$4B_{16}$	113_8	075_{10}	K		
$0100110{0}_2$	$4C_{16}$	114_8	076_{10}	L		
$0100110{1}_2$	$4D_{16}$	115_8	077_{10}	M		
$0100111{0}_2$	$4E_{16}$	116_8	078_{10}	N		
$0100111{1}_2$	$4F_{16}$	117_8	079_{10}	O		
$0101000{0}_2$	50_{16}	120_8	080_{10}	P		
$0101000{1}_2$	51_{16}	121_8	081_{10}	S		
$0101001{0}_2$	52_{16}	122_8	082_{10}	R		
$0101001{1}_2$	53_{16}	123_8	083_{10}	S		
$0101010{0}_2$	54_{16}	124_8	084_{10}	T		
$0101010{1}_2$	55_{16}	125_8	085_{10}	U		
$0101011{0}_2$	56_{16}	126_8	086_{10}	V		
$0101011{1}_2$	57_{16}	127_8	087_{10}	W		

Binario	Esadecimale	Ottale	Decimale	Carattere	<^x>	Linguaggio C
01011000_2	58_{16}	130_8	088_{10}	X		
01011001_2	59_{16}	131_8	089_{10}	Y		
01011010_2	$5A_{16}$	132_8	090_{10}	Z		
01011011_2	$5B_{16}$	133_8	091_{10}	[		
01011100_2	$5C_{16}$	134_8	092_{10}	\		\\
01011101_2	$5D_{16}$	135_8	093_{10}	]		
01011110_2	$5E_{16}$	136_8	094_{10}	^		
01011111_2	$5F_{16}$	137_8	095_{10}	–		
01100000_2	60_{16}	140_8	096_{10}	‘		
01100001_2	61_{16}	141_8	097_{10}	a		
01100010_2	62_{16}	142_8	098_{10}	b		
01100011_2	63_{16}	143_8	099_{10}	c		
01100100_2	64_{16}	144_8	100_{10}	d		
01100101_2	65_{16}	145_8	101_{10}	e		
01100110_2	66_{16}	146_8	102_{10}	f		
01100111_2	67_{16}	147_8	103_{10}	g		
01101000_2	68_{16}	150_8	104_{10}	h		
01101001_2	69_{16}	151_8	105_{10}	i		
01101010_2	$6A_{16}$	152_8	106_{10}	j		
01101011_2	$6B_{16}$	153_8	107_{10}	k		
01101100_2	$6C_{16}$	154_8	108_{10}	l		
01101101_2	$6D_{16}$	155_8	109_{10}	m		

Binario	Esadecimale	Ottale	Decimale	Carattere	<^*x*>	Linguaggio C
01101110_2	$6E_{16}$	156_8	110_{10}	n		
01101111_2	$6F_{16}$	157_8	111_{10}	o		
01110000_2	70_{16}	160_8	112_{10}	p		
01110001_2	71_{16}	161_8	113_{10}	q		
01110010_2	72_{16}	162_8	114_{10}	r		
01110011_2	73_{16}	163_8	115_{10}	s		
01110100_2	74_{16}	164_8	116_{10}	t		
01110101_2	75_{16}	165_8	117_{10}	u		
01110110_2	76_{16}	166_8	118_{10}	v		
01110111_2	77_{16}	167_8	119_{10}	w		
01111000_2	78_{16}	170_8	120_{10}	x		
01111001_2	79_{16}	171_8	121_{10}	y		
01111010_2	$7A_{16}$	172_8	122_{10}	z		
01111011_2	$7B_{16}$	173_8	123_{10}	{		
01111100_2	$7C_{16}$	174_8	124_{10}	|		
01111101_2	$7D_{16}$	175_8	125_{10}	}		
01111110_2	$7E_{16}$	176_8	126_{10}	~		
01111111_2	$7F_{16}$	177_8	127_{10}	*<DEL>*		

47.7.6 ISO 8859-*n*

Le codifiche ISO 8859-***n***, dove ***n*** è un numero da 1 a 15, rappresentano per il passato l'evoluzione più coerente dell'ASCII, in quanto utilizzano tutte gli stessi punti di codifica iniziali da 0 a 127, corrispondenti esattamente all'ASCII originale.

Come nel caso dell'ASCII, non c'è distinzione tra punto di codifica e forma codificata del carattere; in questa situazione si usano valori fino a 255, attraverso un byte intero.

Le codifiche ISO 8859-***n*** introducono altri codici di controllo, nell'intervallo di punti di codifica che va da 128 a 159.

Per quanto riguarda le lingue occidentali, la codifica ISO 8859 più comune è ISO 8859-1, conosciuta anche come ISO Latin 1, rimpiazzata poi da ISO 8859-15 (ISO Latin 9), in cui si inserisce il simbolo dell'Euro al posto del simbolo di valuta generico.

La tabella 47.39 mostra nel dettaglio la codifica ISO 8859-1. Si tenga presente che nel caso di ISO 8859-15, il punto di codifica 164 viene abbinato al simbolo dell'Euro.

Tabella 47.39. ISO 8859-1.

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
240_8	160_{10}	$A0_{16}$		NO-BREAK SPACE
241_8	161_{10}	$A1_{16}$	¡	INVERTED EXCLAMATION MARK
242_8	162_{10}	$A2_{16}$	¢	CENT SIGN
243_8	163_{10}	$A3_{16}$	£	POUND SIGN
244_8	164_{10}	$A4_{16}$	□	CURRENCY SIGN
245_8	165_{10}	$A5_{16}$	¥	YEN SIGN
246_8	166_{10}	$A6_{16}$	¦	BROKEN BAR

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
247_8	167_{10}	$A7_{16}$	§	SECTION SIGN
250_8	168_{10}	$A8_{16}$	¨	DIAERESIS
251_8	169_{10}	$A9_{16}$	©	COPYRIGHT SIGN
252_8	170_{10}	AA_{16}	ª	FEMININE ORDINAL INDICATOR
253_8	171_{10}	AB_{16}	«	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
254_8	172_{10}	AC_{16}	¬	NOT SIGN
255_8	173_{10}	AD_{16}		SOFT HYPHEN
256_8	174_{10}	AE_{16}	®	REGISTERED SIGN
257_8	175_{10}	AF_{16}	¯	MACRON
260_8	176_{10}	$B0_{16}$	°	DEGREE SIGN
261_8	177_{10}	$B1_{16}$	±	PLUS-MINUS SIGN
262_8	178_{10}	$B2_{16}$	²	SUPERSCRIPT TWO
263_8	179_{10}	$B3_{16}$	³	SUPERSCRIPT THREE
264_8	180_{10}	$B4_{16}$	´	ACUTE ACCENT
265_8	181_{10}	$B5_{16}$	μ	MICRO SIGN
266_8	182_{10}	$B6_{16}$	¶	PILCROW SIGN
267_8	183_{10}	$B7_{16}$	·	MIDDLE DOT
270_8	184_{10}	$B8_{16}$	¸	CEDILLA
271_8	185_{10}	$B9_{16}$	¹	SUPERSCRIPT ONE
272_8	186_{10}	BA_{16}	º	MASCULINE ORDINAL INDICATOR
273_8	187_{10}	BB_{16}	»	RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK
274_8	188_{10}	BC_{16}	¼	VULGAR FRACTION ONE QUARTER
275_8	189_{10}	BD_{16}	½	VULGAR FRACTION ONE HALF
276_8	190_{10}	BE_{16}	¾	VULGAR FRACTION THREE QUARTERS
277_8	191_{10}	BF_{16}	¿	INVERTED QUESTION MARK
300_8	192_{10}	$C0_{16}$	À	LATIN CAPITAL LETTER A WITH GRAVE

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
301_8	193_{10}	$C1_{16}$	Á	LATIN CAPITAL LETTER A WITH ACUTE
302_8	194_{10}	$C2_{16}$	Â	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
303_8	195_{10}	$C3_{16}$	Ã	LATIN CAPITAL LETTER A WITH TILDE
304_8	196_{10}	$C4_{16}$	Ä	LATIN CAPITAL LETTER A WITH DIAERESIS
305_8	197_{10}	$C5_{16}$	Å	LATIN CAPITAL LETTER A WITH RING ABOVE
306_8	198_{10}	$C6_{16}$	Æ	LATIN CAPITAL LETTER AE
307_8	199_{10}	$C7_{16}$	Ç	LATIN CAPITAL LETTER C WITH CEDILLA
310_8	200_{10}	$C8_{16}$	È	LATIN CAPITAL LETTER E WITH GRAVE
311_8	201_{10}	$C9_{16}$	É	LATIN CAPITAL LETTER E WITH ACUTE
312_8	202_{10}	CA_{16}	Ê	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
313_8	203_{10}	CB_{16}	Ë	LATIN CAPITAL LETTER E WITH DIAERESIS
314_8	204_{10}	CC_{16}	Ì	LATIN CAPITAL LETTER I WITH GRAVE
315_8	205_{10}	CD_{16}	Í	LATIN CAPITAL LETTER I WITH ACUTE
316_8	206_{10}	CE_{16}	Î	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
317_8	207_{10}	CF_{16}	Ï	LATIN CAPITAL LETTER I WITH DIAERESIS
320_8	208_{10}	$D0_{16}$	Ð	LATIN CAPITAL LETTER ETH
321_8	209_{10}	$D1_{16}$	Ñ	LATIN CAPITAL LETTER N WITH TILDE
322_8	210_{10}	$D2_{16}$	Ò	LATIN CAPITAL LETTER O WITH GRAVE

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
323_8	211_{10}	$D3_{16}$	Ó	LATIN CAPITAL LETTER O WITH ACUTE
324_8	212_{10}	$D4_{16}$	Ô	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
325_8	213_{10}	$D5_{16}$	Õ	LATIN CAPITAL LETTER O WITH TILDE
326_8	214_{10}	$D6_{16}$	Ö	LATIN CAPITAL LETTER O WITH DIAERESIS
327_8	215_{10}	$D7_{16}$	×	MULTIPLICATION SIGN
330_8	216_{10}	$D8_{16}$	Ø	LATIN CAPITAL LETTER O WITH STROKE
331_8	217_{10}	$D9_{16}$	Ù	LATIN CAPITAL LETTER U WITH GRAVE
332_8	218_{10}	DA_{16}	Ú	LATIN CAPITAL LETTER U WITH ACUTE
333_8	219_{10}	DB_{16}	Û	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
334_8	220_{10}	DC_{16}	Ü	LATIN CAPITAL LETTER U WITH DIAERESIS
335_8	221_{10}	DD_{16}	Ý	LATIN CAPITAL LETTER Y WITH ACUTE
336_8	222_{10}	DE_{16}	Þ	LATIN CAPITAL LETTER THORN
337_8	223_{10}	DF_{16}	ß	LATIN SMALL LETTER SHARP S
340_8	224_{10}	$E0_{16}$	à	LATIN SMALL LETTER A WITH GRAVE
341_8	225_{10}	$E1_{16}$	á	LATIN SMALL LETTER A WITH ACUTE
342_8	226_{10}	$E2_{16}$	â	LATIN SMALL LETTER A WITH CIRCUMFLEX
343_8	227_{10}	$E3_{16}$	ã	LATIN SMALL LETTER A WITH TILDE
344_8	228_{10}	$E4_{16}$	ä	LATIN SMALL LETTER A WITH DIAERESIS
345_8	229_{10}	$E5_{16}$	å	LATIN SMALL LETTER A WITH RING ABOVE
346_8	230_{10}	$E6_{16}$	æ	LATIN SMALL LETTER AE

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
347_8	231_{10}	$E7_{16}$	ç	LATIN SMALL LETTER C WITH CEDILLA
350_8	232_{10}	$E8_{16}$	è	LATIN SMALL LETTER E WITH GRAVE
351_8	233_{10}	$E9_{16}$	é	LATIN SMALL LETTER E WITH ACUTE
352_8	234_{10}	EA_{16}	ê	LATIN SMALL LETTER E WITH CIRCUMFLEX
353_8	235_{10}	EB_{16}	ë	LATIN SMALL LETTER E WITH DIAERESIS
354_8	236_{10}	EC_{16}	ì	LATIN SMALL LETTER I WITH GRAVE
355_8	237_{10}	ED_{16}	í	LATIN SMALL LETTER I WITH ACUTE
356_8	238_{10}	EE_{16}	î	LATIN SMALL LETTER I WITH CIRCUMFLEX
357_8	239_{10}	EF_{16}	ï	LATIN SMALL LETTER I WITH DIAERESIS
360_8	240_{10}	$F0_{16}$	ð	LATIN SMALL LETTER ETH
361_8	241_{10}	$F1_{16}$	ñ	LATIN SMALL LETTER N WITH TILDE
362_8	242_{10}	$F2_{16}$	ò	LATIN SMALL LETTER O WITH GRAVE
363_8	243_{10}	$F3_{16}$	ó	LATIN SMALL LETTER O WITH ACUTE
364_8	244_{10}	$F4_{16}$	ô	LATIN SMALL LETTER O WITH CIRCUMFLEX
365_8	245_{10}	$F5_{16}$	õ	LATIN SMALL LETTER O WITH TILDE
366_8	246_{10}	$F6_{16}$	ö	LATIN SMALL LETTER O WITH DIAERESIS
367_8	247_{10}	$F7_{16}$	÷	DIVISION SIGN
370_8	248_{10}	$F8_{16}$	ø	LATIN SMALL LETTER O WITH STROKE
371_8	249_{10}	$F9_{16}$	ù	LATIN SMALL LETTER U WITH GRAVE
372_8	250_{10}	FA_{16}	ú	LATIN SMALL LETTER U WITH ACUTE
373_8	251_{10}	FB_{16}	û	LATIN SMALL LETTER U WITH CIRCUMFLEX
374_8	252_{10}	FC_{16}	ü	LATIN SMALL LETTER U WITH DIAERESIS

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
375_8	253_{10}	FD_{16}	ý	LATIN SMALL LETTER Y WITH ACUTE
376_8	254_{10}	FE_{16}	þ	LATIN SMALL LETTER THORN
377_8	255_{10}	FF_{16}	ÿ	LATIN SMALL LETTER Y WITH DIAERESIS

47.7.7 IBM Code Page *nnn* Character Set

I primi elaboratori IBM con sistema operativo Dos hanno utilizzato delle codifiche particolari, denominate *code page*. Due tra le varie codifiche sono molto importanti: CP 437 e CP 850. La prima era la codifica predefinita, mentre la seconda era quella più utile per le lingue europee occidentali.

Tabella 47.40. Prima parte comune delle codifiche CP 437 e CP 850.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
000_8	0_{10}	00_{16}	U+0000	*<NUL>*
001_8	1_{10}	01_{16}	U+263A	☐
002_8	2_{10}	02_{16}	U+263B	☐
003_8	3_{10}	03_{16}	U+2665	♥
004_8	4_{10}	04_{16}	U+2666	♦
005_8	5_{10}	05_{16}	U+2663	♣
006_8	6_{10}	06_{16}	U+2660	♠
007_8	7_{10}	07_{16}	U+2022	•
010_8	8_{10}	08_{16}	U+25D8	☐
011_8	9_{10}	09_{16}	U+25CB	☐

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
012_8	10_{10}	$0A_{16}$	U+25D9	□
013_8	11_{10}	$0B_{16}$	U+2642	□
014_8	12_{10}	$0C_{16}$	U+2640	□
015_8	13_{10}	$0D_{16}$	U+266A	□
016_8	14_{10}	$0E_{16}$	U+266B	□
017_8	15_{10}	$0F_{16}$	U+263C	□
020_8	16_{10}	10_{16}	U+25B6	□
021_8	17_{10}	11_{16}	U+25C0	□
022_8	18_{10}	12_{16}	U+2195	↕
023_8	19_{10}	13_{16}	U+203C	!!
024_8	20_{10}	14_{16}	U+00B6	¶
025_8	21_{10}	15_{16}	U+00A7	§
026_8	22_{10}	16_{16}	U+25AC	□
027_8	23_{10}	17_{16}	U+21A8	□
030_8	24_{10}	18_{16}	U+2191	↑
031_8	25_{10}	19_{16}	U+2193	v
032_8	26_{10}	$1A_{16}$	U+2192	→
033_8	27_{10}	$1B_{16}$	U+2190	←
034_8	28_{10}	$1C_{16}$	U+221F	□
035_8	29_{10}	$1D_{16}$	U+2194	↔
036_8	30_{10}	$1E_{16}$	U+25B2	▲
037_8	31_{10}	$1F_{16}$	U+25BC	▼
040_8	32_{10}	20_{16}	U+0020	
041_8	33_{10}	21_{16}	U+0021	!
042_8	34_{10}	22_{16}	U+0022	"
043_8	35_{10}	23_{16}	U+0023	#
044_8	36_{10}	24_{16}	U+0024	$

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
045_8	37_{10}	25_{16}	U+0025	%
046_8	38_{10}	26_{16}	U+0026	&
047_8	39_{10}	27_{16}	U+0027	’
050_8	40_{10}	28_{16}	U+0028	(
051_8	41_{10}	29_{16}	U+0029	)
052_8	42_{10}	$2A_{16}$	U+002A	*
053_8	43_{10}	$2B_{16}$	U+002B	+
054_8	44_{10}	$2C_{16}$	U+002C	,
055_8	45_{10}	$2D_{16}$	U+002D	-
056_8	46_{10}	$2E_{16}$	U+002E	.
057_8	47_{10}	$2F_{16}$	U+002F	/
060_8	48_{10}	30_{16}	U+0030	0
061_8	49_{10}	31_{16}	U+0031	1
062_8	50_{10}	32_{16}	U+0032	2
063_8	51_{10}	33_{16}	U+0033	3
064_8	52_{10}	34_{16}	U+0034	4
065_8	53_{10}	35_{16}	U+0035	5
066_8	54_{10}	36_{16}	U+0036	6
067_8	55_{10}	37_{16}	U+0037	7
070_8	56_{10}	38_{16}	U+0038	8
071_8	57_{10}	39_{16}	U+0039	9
072_8	58_{10}	$3A_{16}$	U+003A	:
073_8	59_{10}	$3B_{16}$	U+003B	;
074_8	60_{10}	$3C_{16}$	U+003C	<
075_8	61_{10}	$3D_{16}$	U+003D	=
076_8	62_{10}	$3E_{16}$	U+003E	>
077_8	63_{10}	$3F_{16}$	U+003F	?

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
100_8	64_{10}	40_{16}	U+0040	@
101_8	65_{10}	41_{16}	U+0041	A
102_8	66_{10}	42_{16}	U+0042	B
103_8	67_{10}	43_{16}	U+0043	C
104_8	68_{10}	44_{16}	U+0044	D
105_8	69_{10}	45_{16}	U+0045	E
106_8	70_{10}	46_{16}	U+0046	F
107_8	71_{10}	47_{16}	U+0047	G
110_8	72_{10}	48_{16}	U+0048	H
111_8	73_{10}	49_{16}	U+0049	I
112_8	74_{10}	$4A_{16}$	U+004A	J
113_8	75_{10}	$4B_{16}$	U+004B	K
114_8	76_{10}	$4C_{16}$	U+004C	L
115_8	77_{10}	$4D_{16}$	U+004D	M
116_8	78_{10}	$4E_{16}$	U+004E	N
117_8	79_{10}	$4F_{16}$	U+004F	O
120_8	80_{10}	50_{16}	U+0050	P
121_8	81_{10}	51_{16}	U+0051	Q
122_8	82_{10}	52_{16}	U+0052	R
123_8	83_{10}	53_{16}	U+0053	S
124_8	84_{10}	54_{16}	U+0054	T
125_8	85_{10}	55_{16}	U+0055	U
126_8	86_{10}	56_{16}	U+0056	V
127_8	87_{10}	57_{16}	U+0057	W
130_8	88_{10}	58_{16}	U+0058	X
131_8	89_{10}	59_{16}	U+0059	Y
132_8	90_{10}	$5A_{16}$	U+005A	Z

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
133_8	91_{10}	$5B_{16}$	U+005B	[
134_8	92_{10}	$5C_{16}$	U+005C	\
135_8	93_{10}	$5D_{16}$	U+005D	]
136_8	94_{10}	$5E_{16}$	U+005E	^
137_8	95_{10}	$5F_{16}$	U+005F	–
140_8	96_{10}	60_{16}	U+0060	‘
141_8	97_{10}	61_{16}	U+0061	a
142_8	98_{10}	62_{16}	U+0062	b
143_8	99_{10}	63_{16}	U+0063	c
144_8	100_{10}	64_{16}	U+0064	d
145_8	101_{10}	65_{16}	U+0065	e
146_8	102_{10}	66_{16}	U+0066	f
147_8	103_{10}	67_{16}	U+0067	g
150_8	104_{10}	68_{16}	U+0068	h
151_8	105_{10}	69_{16}	U+0069	i
152_8	106_{10}	$6A_{16}$	U+006A	j
153_8	107_{10}	$6B_{16}$	U+006B	k
154_8	108_{10}	$6C_{16}$	U+006C	l
155_8	109_{10}	$6D_{16}$	U+006D	m
156_8	110_{10}	$6E_{16}$	U+006E	n
157_8	111_{10}	$6F_{16}$	U+006F	o
160_8	112_{10}	70_{16}	U+0070	p
161_8	113_{10}	71_{16}	U+0071	q
162_8	114_{10}	72_{16}	U+0072	r
163_8	115_{10}	73_{16}	U+0073	s
164_8	116_{10}	74_{16}	U+0074	t
165_8	117_{10}	75_{16}	U+0075	u

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
166_8	118_{10}	76_{16}	U+0076	v
167_8	119_{10}	77_{16}	U+0077	w
170_8	120_{10}	78_{16}	U+0078	x
171_8	121_{10}	79_{16}	U+0079	y
172_8	122_{10}	$7A_{16}$	U+007A	z
173_8	123_{10}	$7B_{16}$	U+007B	{
174_8	124_{10}	$7C_{16}$	U+007C	|
175_8	125_{10}	$7D_{16}$	U+007D	}
176_8	126_{10}	$7E_{16}$	U+007E	~
177_8	127_{10}	$7F_{16}$	U+2302	⌂

Tabella 47.41. Seconda parte della codifica CP 437.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
200_8	128_{10}	80_{16}	U+00C7	Ç
201_8	129_{10}	81_{16}	U+00FC	ü
202_8	130_{10}	82_{16}	U+00E9	é
203_8	131_{10}	83_{16}	U+00E2	â
204_8	132_{10}	84_{16}	U+00E4	ä
205_8	133_{10}	85_{16}	U+00E3	ã
206_8	134_{10}	86_{16}	U+00E5	å
207_8	135_{10}	87_{16}	U+00E7	ç
210_8	136_{10}	88_{16}	U+00EA	ê
211_8	137_{10}	89_{16}	U+00EB	ë
212_8	138_{10}	$8A_{16}$	U+00E8	è
213_8	139_{10}	$8B_{16}$	U+00EF	ï

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
214_8	140_{10}	$8C_{16}$	U+00EE	î
215_8	141_{10}	$8D_{16}$	U+00EC	ì
216_8	142_{10}	$8E_{16}$	U+00C4	Ä
217_8	143_{10}	$8F_{16}$	U+00C5	Å
220_8	144_{10}	90_{16}	U+00C9	É
221_8	145_{10}	91_{16}	U+00E6	æ
222_8	146_{10}	92_{16}	U+00C6	Æ
223_8	147_{10}	93_{16}	U+00F4	ô
224_8	148_{10}	94_{16}	U+00F6	ö
225_8	149_{10}	95_{16}	U+00F2	ò
226_8	150_{10}	96_{16}	U+00FB	û
227_8	151_{10}	97_{16}	U+00F9	ù
230_8	152_{10}	98_{16}	U+00FF	ÿ
231_8	153_{10}	99_{16}	U+00D6	Ö
232_8	154_{10}	$9A_{16}$	U+00DC	Ü
233_8	155_{10}	$9B_{16}$	U+00F8	ø
234_8	156_{10}	$9C_{16}$	U+00A3	£
235_8	157_{10}	$9D_{16}$	U+00A5	¥
236_8	158_{10}	$9E_{16}$	U+20A7	□
237_8	159_{10}	$9F_{16}$	U+0192	*f*
240_8	160_{10}	$A0_{16}$	U+00E1	á
241_8	161_{10}	$A1_{16}$	U+00ED	í
242_8	162_{10}	$A2_{16}$	U+00F3	ó
243_8	163_{10}	$A3_{16}$	U+00FA	ú
244_8	164_{10}	$A4_{16}$	U+00F1	ñ
245_8	165_{10}	$A5_{16}$	U+00D1	Ñ
246_8	166_{10}	$A6_{16}$	U+00AA	ª

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
247_8	167_{10}	$A7_{16}$	U+00BA	º
250_8	168_{10}	$A8_{16}$	U+00BF	¿
251_8	169_{10}	$A9_{16}$	U+00AE	®
252_8	170_{10}	AA_{16}	U+00AC	¬
253_8	171_{10}	AB_{16}	U+00BD	½
254_8	172_{10}	AC_{16}	U+00BC	¼
255_8	173_{10}	AD_{16}	U+00A1	¡
256_8	174_{10}	AE_{16}	U+00AB	«
257_8	175_{10}	AF_{16}	U+00BB	»
260_8	176_{10}	$B0_{16}$	U+2591	□
261_8	177_{10}	$B1_{16}$	U+2592	□
262_8	178_{10}	$B2_{16}$	U+2593	□
263_8	179_{10}	$B3_{16}$	U+2502	|
264_8	180_{10}	$B4_{16}$	U+2524	|
265_8	181_{10}	$B5_{16}$	U+2561	□
266_8	182_{10}	$B6_{16}$	U+2562	□
267_8	183_{10}	$B7_{16}$	U+2556	□
270_8	184_{10}	$B8_{16}$	U+2555	□
271_8	185_{10}	$B9_{16}$	U+2563	□
272_8	186_{10}	BA_{16}	U+2551	□
273_8	187_{10}	BB_{16}	U+2557	□
274_8	188_{10}	BC_{16}	U+255D	□
275_8	189_{10}	BD_{16}	U+255C	□
276_8	190_{10}	BE_{16}	U+255B	□
277_8	191_{10}	BF_{16}	U+2510	·
300_8	192_{10}	$C0_{16}$	U+2514	‘
301_8	193_{10}	$C1_{16}$	U+2534	□

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
302_8	194_{10}	$C2_{16}$	U+252C	□
303_8	195_{10}	$C3_{16}$	U+251C	|
304_8	196_{10}	$C4_{16}$	U+2500	-
305_8	197_{10}	$C5_{16}$	U+253C	□
306_8	198_{10}	$C6_{16}$	U+255E	□
307_8	199_{10}	$C7_{16}$	U+255F	□
310_8	200_{10}	$C8_{16}$	U+255A	□
311_8	201_{10}	$C9_{16}$	U+2554	□
312_8	202_{10}	CA_{16}	U+2569	□
313_8	203_{10}	CB_{16}	U+2566	□
314_8	204_{10}	CC_{16}	U+2560	□
315_8	205_{10}	CD_{16}	U+2550	□
316_8	206_{10}	CE_{16}	U+256C	□
317_8	207_{10}	CF_{16}	U+2567	□
320_8	208_{10}	$D0_{16}$	U+2568	□
321_8	209_{10}	$D1_{16}$	U+2564	□
322_8	210_{10}	$D2_{16}$	U+2565	□
323_8	211_{10}	$D3_{16}$	U+2559	□
324_8	212_{10}	$D4_{16}$	U+2558	□
325_8	213_{10}	$D5_{16}$	U+2552	□
326_8	214_{10}	$D6_{16}$	U+2553	□
327_8	215_{10}	$D7_{16}$	U+256B	□
330_8	216_{10}	$D8_{16}$	U+256A	□
331_8	217_{10}	$D9_{16}$	U+2518	’
332_8	218_{10}	DA_{16}	U+250C	·
333_8	219_{10}	DB_{16}	U+2588	□
334_8	220_{10}	DC_{16}	U+2584	□

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
335_8	221_{10}	DD_{16}	U+258C	□
336_8	222_{10}	DE_{16}	U+2590	□
337_8	223_{10}	DF_{16}	U+2580	□
340_8	224_{10}	$E0_{16}$	U+03B1	α
341_8	225_{10}	$E1_{16}$	U+00DF	ß
342_8	226_{10}	$E2_{16}$	U+0393	Γ
343_8	227_{10}	$E3_{16}$	U+03C0	π
344_8	228_{10}	$E4_{16}$	U+03A3	Σ
345_8	229_{10}	$E5_{16}$	U+03C3	σ
346_8	230_{10}	$E6_{16}$	U+00B5	μ
347_8	231_{10}	$E7_{16}$	U+03C4	τ
350_8	232_{10}	$E8_{16}$	U+03A6	Φ
351_8	233_{10}	$E9_{16}$	U+0398	Θ
352_8	234_{10}	EA_{16}	U+03A9	Ω
353_8	235_{10}	EB_{16}	U+03B4	δ
354_8	236_{10}	EC_{16}	U+221E	∞
355_8	237_{10}	ED_{16}	U+03C6	ϕ
356_8	238_{10}	EE_{16}	U+03B5	ε
357_8	239_{10}	EF_{16}	U+2229	∩
360_8	240_{10}	$F0_{16}$	U+2261	≡
361_8	241_{10}	$F1_{16}$	U+00B1	±
362_8	242_{10}	$F2_{16}$	U+2265	≥
363_8	243_{10}	$F3_{16}$	U+2264	≤
364_8	244_{10}	$F4_{16}$	U+2320	⌠
365_8	245_{10}	$F5_{16}$	U+2321	⌡
366_8	246_{10}	$F6_{16}$	U+00F7	÷
367_8	247_{10}	$F7_{16}$	U+2248	≈

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
370_8	248_{10}	$F8_{16}$	U+00B0	°
371_8	249_{10}	$F9_{16}$	U+2219	∙
372_8	250_{10}	FA_{16}	U+00B7	·
373_8	251_{10}	FB_{16}	U+221A	√
374_8	252_{10}	FC_{16}	U+207F	^{n}
375_8	253_{10}	FD_{16}	U+00B2	^{2}
376_8	254_{10}	FE_{16}	U+25A0	■
377_8	255_{10}	FF_{16}	U+00A0	

Tabella 47.42. Seconda parte della codifica CP 850.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
200_8	128_{10}	80_{16}	U+00C7	Ç
201_8	129_{10}	81_{16}	U+00FC	ü
202_8	130_{10}	82_{16}	U+00E9	é
203_8	131_{10}	83_{16}	U+00E2	â
204_8	132_{10}	84_{16}	U+00E4	ä
205_8	133_{10}	85_{16}	U+00E3	ã
206_8	134_{10}	86_{16}	U+00E5	å
207_8	135_{10}	87_{16}	U+00E7	ç
210_8	136_{10}	88_{16}	U+00EA	ê
211_8	137_{10}	89_{16}	U+00EB	ë
212_8	138_{10}	$8A_{16}$	U+00E8	è
213_8	139_{10}	$8B_{16}$	U+00EF	ï
214_8	140_{10}	$8C_{16}$	U+00EE	î
215_8	141_{10}	$8D_{16}$	U+00EC	ì

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
216_8	142_{10}	$8E_{16}$	U+00C4	Ä
217_8	143_{10}	$8F_{16}$	U+00C5	Å
220_8	144_{10}	90_{16}	U+00C9	É
221_8	145_{10}	91_{16}	U+00E6	æ
222_8	146_{10}	92_{16}	U+00C6	Æ
223_8	147_{10}	93_{16}	U+00F4	ô
224_8	148_{10}	94_{16}	U+00F6	ö
225_8	149_{10}	95_{16}	U+00F2	ò
226_8	150_{10}	96_{16}	U+00FB	û
227_8	151_{10}	97_{16}	U+00F9	ù
230_8	152_{10}	98_{16}	U+00FF	ÿ
231_8	153_{10}	99_{16}	U+00D6	Ö
232_8	154_{10}	$9A_{16}$	U+00DC	Ü
233_8	155_{10}	$9B_{16}$	U+00F8	ø
234_8	156_{10}	$9C_{16}$	U+00A3	£
235_8	157_{10}	$9D_{16}$	U+00D8	Ø
236_8	158_{10}	$9E_{16}$	U+00D7	×
237_8	159_{10}	$9F_{16}$	U+0192	*f*
240_8	160_{10}	$A0_{16}$	U+00E1	á
241_8	161_{10}	$A1_{16}$	U+00ED	í
242_8	162_{10}	$A2_{16}$	U+00F3	ó
243_8	163_{10}	$A3_{16}$	U+00FA	ú
244_8	164_{10}	$A4_{16}$	U+00F1	ñ
245_8	165_{10}	$A5_{16}$	U+00D1	Ñ
246_8	166_{10}	$A6_{16}$	U+00AA	ª
247_8	167_{10}	$A7_{16}$	U+00BA	º
250_8	168_{10}	$A8_{16}$	U+00BF	¿

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
251_8	169_{10}	$A9_{16}$	U+00AE	®
252_8	170_{10}	AA_{16}	U+00AC	¬
253_8	171_{10}	AB_{16}	U+00BD	½
254_8	172_{10}	AC_{16}	U+00BC	¼
255_8	173_{10}	AD_{16}	U+00A1	¡
256_8	174_{10}	AE_{16}	U+00AB	«
257_8	175_{10}	AF_{16}	U+00BB	»
260_8	176_{10}	$B0_{16}$	U+2591	□
261_8	177_{10}	$B1_{16}$	U+2592	□
262_8	178_{10}	$B2_{16}$	U+2593	□
263_8	179_{10}	$B3_{16}$	U+2502	|
264_8	180_{10}	$B4_{16}$	U+2524	|
265_8	181_{10}	$B5_{16}$	U+00C1	Á
266_8	182_{10}	$B6_{16}$	U+00C1	Á
267_8	183_{10}	$B7_{16}$	U+00C0	À
270_8	184_{10}	$B8_{16}$	U+00A9	©
271_8	185_{10}	$B9_{16}$	U+2563	□
272_8	186_{10}	BA_{16}	U+2551	□
273_8	187_{10}	BB_{16}	U+2557	□
274_8	188_{10}	BC_{16}	U+255D	□
275_8	189_{10}	BD_{16}	U+00A2	¢
276_8	190_{10}	BE_{16}	U+00A5	¥
277_8	191_{10}	BF_{16}	U+2510	·
300_8	192_{10}	$C0_{16}$	U+2514	‘
301_8	193_{10}	$C1_{16}$	U+2534	□
302_8	194_{10}	$C2_{16}$	U+252C	□
303_8	195_{10}	$C3_{16}$	U+251C	|

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
304_8	196_{10}	$C4_{16}$	U+2500	-
305_8	197_{10}	$C5_{16}$	U+253C	□
306_8	198_{10}	$C6_{16}$	U+00E3	ã
307_8	199_{10}	$C7_{16}$	U+00C3	Ã
310_8	200_{10}	$C8_{16}$	U+255A	□
311_8	201_{10}	$C9_{16}$	U+2554	□
312_8	202_{10}	CA_{16}	U+2569	□
313_8	203_{10}	CB_{16}	U+2566	□
314_8	204_{10}	CC_{16}	U+2560	□
315_8	205_{10}	CD_{16}	U+2550	□
316_8	206_{10}	CE_{16}	U+256C	□
317_8	207_{10}	CF_{16}	U+00A4	□
320_8	208_{10}	$D0_{16}$	U+00F0	ð
321_8	209_{10}	$D1_{16}$	U+00D0	Ð
322_8	210_{10}	$D2_{16}$	U+00CA	Ê
323_8	211_{10}	$D3_{16}$	U+00CB	Ë
324_8	212_{10}	$D4_{16}$	U+00C8	È
325_8	213_{10}	$D5_{16}$	U+0131	ı
326_8	214_{10}	$D6_{16}$	U+00CD	Í
327_8	215_{10}	$D7_{16}$	U+00CE	Î
330_8	216_{10}	$D8_{16}$	U+00CF	Ï
331_8	217_{10}	$D9_{16}$	U+2518	’
332_8	218_{10}	DA_{16}	U+250C	·
333_8	219_{10}	DB_{16}	U+2588	□
334_8	220_{10}	DC_{16}	U+2584	□
335_8	221_{10}	DD_{16}	U+00A6	¦
336_8	222_{10}	DE_{16}	U+00CC	Ì

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
337_8	223_{10}	DF_{16}	U+2580	□
340_8	224_{10}	$E0_{16}$	U+00D3	Ó
341_8	225_{10}	$E1_{16}$	U+00DF	ß
342_8	226_{10}	$E2_{16}$	U+00D4	Ô
343_8	227_{10}	$E3_{16}$	U+00D2	Ò
344_8	228_{10}	$E4_{16}$	U+00F5	õ
345_8	229_{10}	$E5_{16}$	U+00D5	Õ
346_8	230_{10}	$E6_{16}$	U+00B5	μ
347_8	231_{10}	$E7_{16}$	U+00FE	þ
350_8	232_{10}	$E8_{16}$	U+00DE	Þ
351_8	233_{10}	$E9_{16}$	U+00DA	Ú
352_8	234_{10}	EA_{16}	U+00DB	Û
353_8	235_{10}	EB_{16}	U+00D9	Ù
354_8	236_{10}	EC_{16}	U+00FD	ý
355_8	237_{10}	ED_{16}	U+00DD	Ý
356_8	238_{10}	EE_{16}	U+00AF	¯
357_8	239_{10}	EF_{16}	U+00B4	´
360_8	240_{10}	$F0_{16}$	U+00AD	
361_8	241_{10}	$F1_{16}$	U+00B1	±
362_8	242_{10}	$F2_{16}$	U+2017	□
363_8	243_{10}	$F3_{16}$	U+00BE	¾
364_8	244_{10}	$F4_{16}$	U+00B6	¶
365_8	245_{10}	$F5_{16}$	U+00A7	§
366_8	246_{10}	$F6_{16}$	U+00F7	÷
367_8	247_{10}	$F7_{16}$	U+00B8	¸
370_8	248_{10}	$F8_{16}$	U+00B0	°
371_8	249_{10}	$F9_{16}$	U+00A8	¨

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
372_8	250_{10}	FA_{16}	U+00B7	·
373_8	251_{10}	FB_{16}	U+00B9	1
374_8	252_{10}	FC_{16}	U+00B3	3
375_8	253_{10}	FD_{16}	U+00B2	2
376_8	254_{10}	FE_{16}	U+25A0	■
377_8	255_{10}	FF_{16}	U+00A0	

47.7.8 Utilizzo di «ascii»

«

Il programma '**`ascii`**'[15] consente di ottenere facilmente informazioni su un certo carattere appartenente alla codifica ASCII, ovvero ai primi 127 punti di codifica.

`ascii` [*opzioni*] [*carattere*]…

Generalmente, si utilizza il programma specificando nella riga di comando uno o più caratteri, di cui si vogliono maggiori informazioni. I caratteri possono essere indicati letteralmente (ammesso che ciò sia possibile), oppure attraverso una forma alternativa, purché sia stata prevista.

Tabella 47.43. Alcune modalità con cui si può indicare un carattere nella riga di comando.

Forma	Descrizione
x	Se possibile, il carattere può essere indicato in forma letterale.

Forma	Descrizione
`^`***x***	La maggior parte dei caratteri di controllo può essere specificata secondo la forma '**^*x***'.
`\`***x***	Alcuni caratteri di controllo possono essere rappresentati in questo modo, che si rifà alle convenzioni del linguaggio C.
nome	I caratteri di controllo possono essere indicati per nome.
`0x`***n*** `x`***n*** `\x`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in esadecimale.
`0d`***n*** `d`***n*** `\d`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in decimale.
`0o`***n*** `o`***n*** `\o`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in ottale.
`0b`***n*** `b`***n*** `\b`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in binario.

Le opzioni della riga di comando non vengono descritte; si ve-

da eventualmente la pagina di manuale *ascii(1)*. Vengono mostrati alcuni esempi.

```
$ ascii [Invio]
```

In questo modo si ottiene una guida rapida all'utilizzo del programma, assieme al riepilogo della tabella ASCII. Si osservi che si ottiene lo stesso risultato con l'opzione '**-h**'.

```
Usage: ascii [-dxohv] [-t] [char-alias...]
   -t = one-line output  -d = Decimal table  -o = octal table
   -x = hex table -h = This help screen
   -v = version information
Prints all aliases of an ASCII character. Args may be chars,
C \-escapes, English names, ^-escapes, ASCII mnemonics, or
numerics in decimal/octal/hex.

Dec Hex    Dec Hex    Dec Hex  Dec Hex   Dec Hex   Dec Hex
  0 00 NUL  22 16 SYN  44 2C ,  66 42 B   88 58 X  110 6E n
  1 01 SOH  23 17 ETB  45 2D -  67 43 C   89 59 Y  111 6F o
  2 02 STX  24 18 CAN  46 2E .  68 44 D   90 5A Z  112 70 p
  3 03 ETX  25 19 EM   47 2F /  69 45 E   91 5B [  113 71 q
  4 04 EOT  26 1A SUB  48 30 0  70 46 F   92 5C \  114 72 r
  5 05 ENQ  27 1B ESC  49 31 1  71 47 G   93 5D ]  115 73 s
  6 06 ACK  28 1C FS   50 32 2  72 48 H   94 5E ^  116 74 t
  7 07 BEL  29 1D GS   51 33 3  73 49 I   95 5F _  117 75 u
  8 08 BS   30 1E RS   52 34 4  74 4A J   96 60 `  118 76 v
  9 09 HT   31 1F US   53 35 5  75 4B K   97 61 a  119 77 w
 10 0A LF   32 20      54 36 6  76 4C L   98 62 b  120 78 x
 11 0B VT   33 21 !    55 37 7  77 4D M   99 63 c  121 79 y
 12 0C FF   34 22 "    56 38 8  78 4E N  100 64 d  122 7A z
 13 0D CR   35 23 #    57 39 9  79 4F O  101 65 e  123 7B {
 14 0E SO   36 24 $    58 3A :  80 50 P  102 66 f  124 7C |
 15 0F SI   37 25 %    59 3B ;  81 51 Q  103 67 g  125 7D }
 16 10 DLE  38 26 &    60 3C <  82 52 R  104 68 h  126 7E ~
 17 11 DC1  39 27 '    61 3D =  83 53 S  105 69 i  127 7F DEL
```

```
 18 12 DC2  40 28 (    62 3E >  84 54 T  106 6A j
 19 13 DC3  41 29 )    63 3F ?  85 55 U  107 6B k
 20 14 DC4  42 2A *    64 40 @  86 56 V  108 6C l
 21 15 NAK  43 2B +    65 41 A  87 57 W  109 6D m
```

$ **ascii ETX**[*Invio*]

L'argomento rappresenta il nome di un codice di controllo secondo le convenzioni della codifica ASCII. Si può osservare che questo carattere si può rappresentare, convenzionalmente, anche con la sequenza <^*C*>.

```
ASCII 0/3 is decimal 003, hex 03, octal 003, bits 00000011:
called ^C, ETX Official name: End of Text
```

$ **ascii ^c**[*Invio*]

Si ottiene esattamente lo stesso risultato dell'esempio precedente.

$ **ascii a b c**[*Invio*]

Mostra le informazioni sui tre caratteri inseriti:

```
ASCII 6/1 is decimal 097, hex 61, octal 141, bits 01100001:
prints as 'a'
Official name: Miniscule a
Other names: Small a, Lowercase a

ASCII 6/2 is decimal 098, hex 62, octal 142, bits 01100010:
prints as 'b'
Official name: Miniscule b
Other names: Small b, Lowercase b

ASCII 6/3 is decimal 099, hex 63, octal 143, bits 01100011:
prints as 'c'
Official name: Miniscule c
```

```
  Other names: Small c, Lowercase c
```

```
$ ascii -t a b c[Invio]
```

Mostra alcune informazioni sui caratteri indicati come argomento finale:

```
  6/1   97   0x61   0o141   01100001
  6/2   98   0x62   0o142   01100010
  6/3   99   0x63   0o143   01100011
```

```
$ ascii -s abc[Invio]
```

Mostra alcune informazioni sui caratteri che compongono la stringa fornita come argomento finale:

```
  6/1   97   0x61   0o141   01100001
  6/2   98   0x62   0o142   01100010
  6/3   99   0x63   0o143   01100011
```

47.7.9 Utilizzo di «unicode»

Il programma '**unicode**'[16] consente di ottenere facilmente informazioni su un certo carattere, secondo la classificazione della codifica universale.

```
unicode [opzioni] [carattere_o_stringa] ...
```

Gli argomenti finali della riga di comando possono essere caratteri singoli, oppure stringhe; l'indicazione di una stringa implica la richiesta di avere informazioni su tutti i caratteri che contiene. I caratteri cercati possono essere indicati letteralmente, oppure attraverso altre forme, come documentato nella pagina di manuale *unicode(1)*,

ma generalmente conviene limitarsi all'uso della notazione tipica per Unicode: '**U+*n***', dove ***n*** è un numero del punto di codifica espresso in esadecimale.

Tabella 47.49. Alcune modalità con cui si possono indicare i caratteri nella riga di comando.

Forma	Descrizione
x	Se possibile, il carattere può essere indicato in forma letterale.
U+***n*** ***n***	Rappresenta il punto di codifica in esadecimale.
U+***n*** **..** U+***m*** ***n*** **..** ***m***	Rappresenta un intervallo di punti di codifica e si usa per ottenere la tabella dei codici relativi.

Sono disponibili delle opzioni, che generalmente non è necessario utilizzare; eventualmente si può consultare la pagina di manuale *unicode(1)*. Vengono mostrati alcuni esempi.

```
$ unicode à[Invio]
```

Richiede informazioni sulla lettera «à» (a minuscola con accento grave).

```
U+00E0 LATIN SMALL LETTER A WITH GRAVE
UTF-8: c3 a0   UTF-16BE: 00e0   Decimal: &#224;
à (À)
Uppercase: U+00C0
Category: Ll (Letter, Lowercase)
Bidi: L (Left-to-Right)
Decomposition: 0061 0300
```

```
$ unicode U+0416 [Invio]
```

Richiede informazioni sul punto di codifica U+0416:

```
U+0416 CYRILLIC CAPITAL LETTER ZHE
UTF-8: d0 96   UTF-16BE: 0416   Decimal: &#1046;
Ж (ж)
Lowercase: U+0436
Category: Lu (Letter, Uppercase)
Bidi: L (Left-to-Right)
```

```
$ unicode 0416 [Invio]
```

Esattamente come nell'esempio precedente.

```
$ unicode U+0005..U+0010 [Invio]
```

Richiede di visualizzare il blocco di codici che comprende i punti di codifica da U+0005 a U+0010; in pratica, visualizza i punti di codifica da U+0000 a U+00FF:

```
      .0 .1 .2 .3 .4 .5 .6 .7 .8 .9 .A .B .C .D .E .F
000.
001.
002.     !  "  #  $  %  &  ’  (  )  *  +  ,  -  .  /
003.  0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
004.  @  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O
005.  P  Q  R  S  T  U  V  W  X  Y  Z  [  \  ]  ^  _
006.  `  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o
007.  p  q  r  s  t  u  v  w  x  y  z  {  |  }  ~
008.
009.
00A.     ¡  ¢  £  □  ¥  ¦  §  ¨  ©  ª  «  ¬     ®  ¯
00B.  °  ±  ²  ³  ´  µ  ¶  ·  ¸  ¹  º  »  ¼  ½  ¾  ¿
00C.  À  Á  Â  Ã  Ä  Å  Æ  Ç  È  É  Ê  Ë  Ì  Í  Î  Ï
00D.  Ð  Ñ  Ò  Ó  Ô  Õ  Ö  ×  Ø  Ù  Ú  Û  Ü  Ý  Þ  ß
```

```
00E.  à  á  â  ã  ä  å  æ  ç  è  é  ê  ë  ì  í  î  ï
00F.  ð  ñ  ò  ó  ô  õ  ö  ÷  ø  ù  ú  û  ü  ý  þ  ÿ
```

47.7.10 Gucharmap

«

Attraverso il sistema grafico X è possibile utilizzare Gucharmap,[17] con il quale si ottengono informazioni sui codici dei caratteri, secondo la codifica universale, in modo pratico.

```
gucharmap
```

Il programma '**gucharmap**' si avvia senza opzioni particolari e si interagisce con lui, attraverso un'interfaccia grafica.

Figura 47.53. Gucharmap con la selezione del punto di codifica U+0D24.

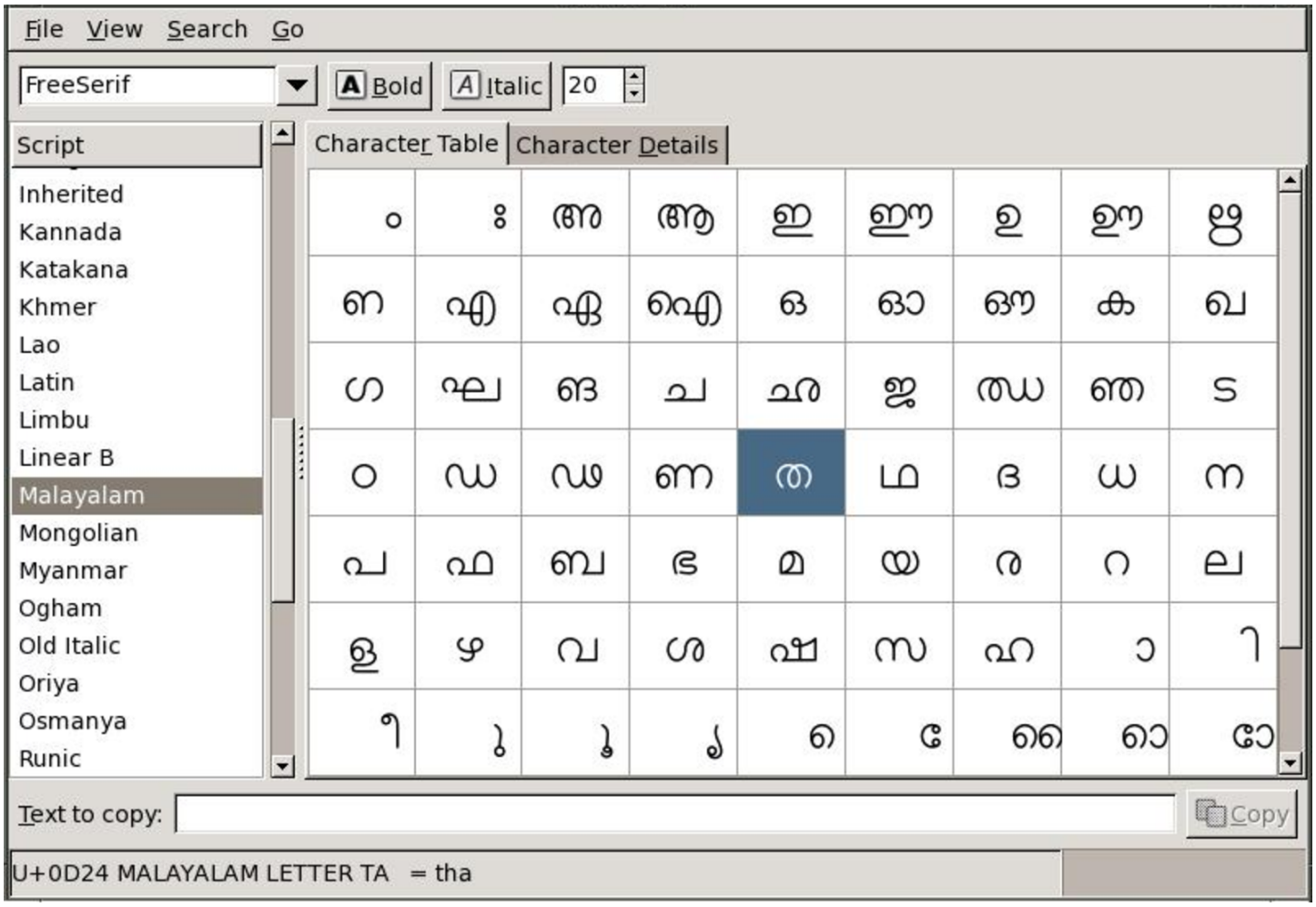
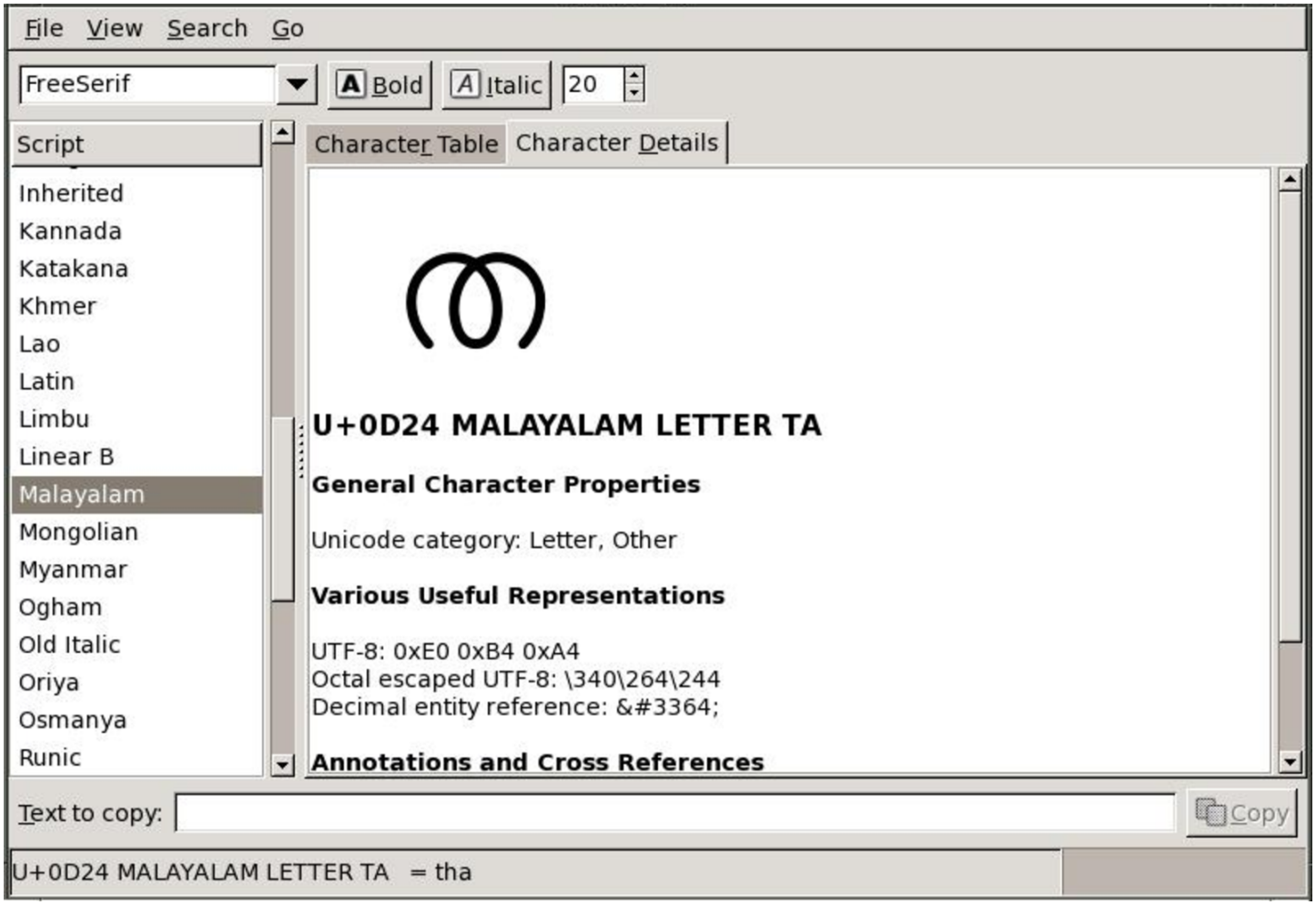


Figura 47.54. Gucharmap con le informazioni dettagliate del punto di codifica U+0D24.

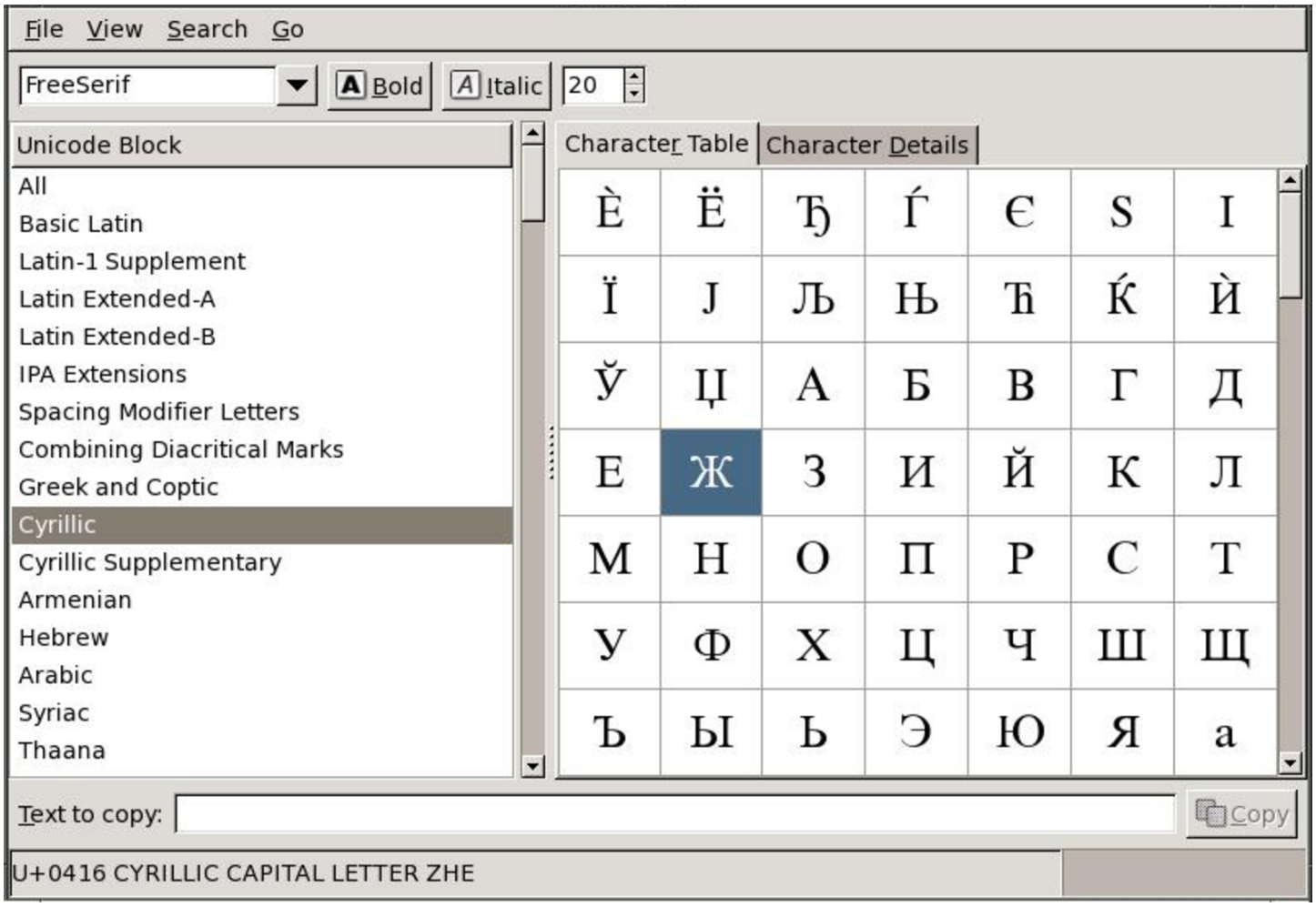


Una volta selezionato il gruppo di codici di proprio interesse, è facile copiare i caratteri desiderati trascinandoli nella riga di testo che appare nella parte inferiore della finestra, come si vede nella figura 47.55. Da lì è poi possibile copiarli nell'applicazione di proprio interesse, utilizzando i metodi consueti per le operazioni di «taglia-copia-incolla».

Figura 47.55. Copia di alcuni caratteri nella riga inferiore.

I caratteri della codifica universale possono essere classificati in base alla scrittura (*script*), oppure in base ai blocchi utilizzati nella codifica stessa. La figura 47.56 mostra una porzione del blocco cirillico (da U+0400 a U+04FF).

Figura 47.56. Classificazione dei codici secondo i blocchi della codifica universale.



47.8 Trasformazione della codifica

La presenza di codifiche diverse comporta spesso la necessità di trasformare i file di testo da una codifica a un'altra, oppure di far lavorare un programma in un contesto che utilizza una codifica diversa da quella impostata generalmente nel sistema.

47.8.1 Recode

Recode[18] è un programma per la conversione di file da un insieme di caratteri a un altro. Recode non si limita semplicemente a questo; spesso è in grado di intervenire su codifiche composte da sequenze di caratteri, anche se in queste situazioni il suo utilizzo si complica e i risultati non sono sempre garantiti.

```
recode [opzioni] codifica_prima..codifica_dopo [file…]
```

Osservando lo schema sintattico mostrato, si può vedere che è necessario indicare il tipo di conversione attraverso due parole chiave: la prima serve a stabilire il modo in cui sono codificati i dati in ingresso, la seconda stabilisce in che modo li si vuole trasformare in uscita.

Uno o più file in ingresso possono essere indicati alla fine della riga di comando, facendo sì che Recode tenti di sovrascriverli; in alternativa, un file in ingresso può essere fornito attraverso lo standard input, ottenendo il risultato della conversione dallo standard output.

Bisogna essere prudenti con Recode quando si indicano i file nella riga di comando, perché la conversione potrebbe essere irreversibile.

Nel suo piccolo, Recode è un programma complesso. Questa sezione mostra solo alcuni aspetti banali, mentre per sfruttare bene tutte le sue potenzialità è necessario leggere la documentazione originale: *info recode*.

Tabella 47.57. Alcune opzioni.

Opzione	Descrizione
`-l` [*codifica*] `--list`[=*codifica*]	Questa opzione, usata senza argomento, mostra l'elenco delle codifiche gestite; con l'aggiunta del nome di una codifica, si ottiene invece una tabella della codifica stessa.
`-g` `--graphics`	Questa opzione riguarda la conversione dall'insieme di caratteri '**`IBM-PC`**', a un altro tipo, dove si vuole tentare di trasformare in qualche modo i simboli grafici tipici di quella codifica. È evidente che questa conversione è irreversibile.

Le codifiche da utilizzare nelle conversioni sono indicate attraverso la notazione ***codifica_prima..codifica_dopo***, come si vede nello schema sintattico introduttivo. Le parole chiave utilizzate per questo possono essere indicate indifferentemente utilizzando le lettere minuscole o maiuscole. L'elenco delle codifiche (e quindi delle trasformazioni possibili) è molto lungo e potrebbe essere ottenuto un riepilogo attraverso l'opzione '**`-l`**' da sola. Tuttavia, sarebbe meglio leggere prima ciò che è stato annotato nel documento *info recode* al riguardo, per non rischiare di trovarsi poi nei pasticci.

Tabella 47.58. Alcune parole chiave che si possono usare per individuare le codifiche.

Codifica	Descrizione

`IBM437`
`437`
`cp437`
	Rappresenta la codifica IBM usata normalmente nel Dos. Quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono lasciati inalterati.

`IBM-PC` `ibmpc`	È praticamente la stessa codifica IBM437, con la differenza che quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono trasformati.

`IBM850`
`850`
`cp850`
	Rappresenta la codifica IBM usata normalmente nel Dos per la localizzazione europea.

Codifica	Descrizione

`ISO_8859-1:1987`
`ISO_8859-1`
`ISO-8859-1`
`CP819`
`IBM819`
`iso-ir-100`
`l1`
`latin1`
	Si riferisce alla codifica ISO 8859-1.

`UTF-8` `UTF8`	Si riferisce alla codifica UTF-8.

Segue la descrizione di alcuni esempi.

`$` **`recode -a IBM437`** [*Invio*]

Mostra tutte le possibilità di abbinamento con la codifica IBM437.

`$` **`recode -a IBM-PC`** [*Invio*]

Mostra tutte le possibilità di abbinamento con la codifica IBM-PC.

`$` **`recode IBM-PC..ISO_8859-1 lettera`** [*Invio*]

Converte il file '`lettera`' dalla codifica IBM-PC, con codice di interruzione di riga pari a <*CR*><*LF*> a ISO 8859-1, con codi-

ce di interruzione di riga pari a <*LF*> soltanto, sovrascrivendo il file.

```
$ recode IBM-PC..ISO_8859-1 < lettera > lettera2 [Invio]
```

Converte il file ‘`lettera`’ dalla codifica ISO 8859-1, con codice di interruzione di riga pari a <*LF*>, a IBM-PC, con codice di interruzione di riga pari a <*CR*><*LF*>, generando il file ‘`lettera2`’.

```
$ recode -g IBM-PC..ISO_8859-1 < schema1 > schema2 [Invio]
```

Converte il file ‘`schema1`’ dalla codifica IBM-PC a ISO 8859-1, generando il file ‘`schema2`’, tentando di convertire anche i simboli grafici. Anche in questo caso ha luogo la conversione del codice di interruzione di riga.

47.8.2 Iconv

Iconv[19] è un programma previsto dallo standard dei sistemi Unix e serve a convertire un file da un insieme di caratteri a un altro. Lo schema sintattico seguente mostra l’utilizzo normale del programma:

```
iconv -f da_codifica -t a_codifica [altre_opzioni] [file...]
```

La codifica viene indicata attraverso una parola chiave e l’elenco completo degli insiemi di caratteri conosciuti si ottiene con l’opzione ‘**-l**’, usata da sola:

```
iconv -l
```

Il file da convertire viene indicato normalmente alla fine della riga di comando, ma in sua mancanza viene usato lo standard input; generalmente il risultato della conversione viene emesso attraverso lo standard output, a meno che sia usata espressamente l'opzione '**-o**', con la quale si indica il nome di un file da creare per questo scopo.

Tabella 47.59. Alcune opzioni.

Opzione	Descrizione
`-f` ***codifica_di_partenza*** `--from-code` ***codifica_di_partenza***	Specifica la codifica di origine.
`-t` ***codifica_di_destinazione*** `--to-code` ***codifica_di_destinazione***	Specifica la codifica del file da generare.
`-l` `--list`	Elenca le parole chiave riconosciute che individuano le varie codifiche gestibili dal programma. Si usa questa opzione da sola.
`-o` ***file_da_generare*** `--output` ***file_da_generare***	Richiede di fornire il risultato nel file specificato, senza usare per questo lo standard output.

Le parole chiave utilizzate per individuare la codifica possono essere indicate indifferentemente utilizzando le lettere minuscole o maiuscole. Si deve osservare che non tutte le trasformazioni sono possibili; pertanto, di fronte a richieste impossibili, il programma si rifiuta di procedere, così come si rifiuta di farlo se il file in ingresso non è conforme alla codifica dichiarata per questo.

Tabella 47.60. Alcune parole chiave che si possono usare per individuare le codifiche.

Codifica	Descrizione

`IBM437`
`437`
`CP437`
	Rappresenta la codifica IBM usata normalmente nel Dos.

`IBM850`
`850`
`CP850`
	Rappresenta la codifica IBM usata normalmente nel Dos per la localizzazione europea.

`LATIN1`
`8859_1`
`ISO-8859-1`
`ISO_8859-1`
`ISO_8859-1:1987`
`ISO8859-1`
`ISO88591`
	Si riferisce alla codifica ISO 8859-1.

`UTF-8` `UTF8`	Si riferisce alla codifica UTF-8.

Segue la descrizione di alcuni esempi.

`$` **`iconv --list`** [*Invio*]

Mostra i nomi di tutte le codifiche gestibili.

`$` **`iconv -f LATIN1 -t UTF-8 lettera > lettera2`** [*Invio*]

Converte il file '`lettera`', scritto con la codifica ISO 8859-1, a UTF-8, generando il file '`lettera2`'.

`$` **`cat lettera | iconv -f LATIN1 -t UTF-8 > lettera2`** [*Invio*]

Esattamente come nell'esempio precedente, con la differenza che il file '`lettera`' viene fornito attraverso lo standard input.

`$` **`iconv -f BIG5 -t UTF-8 lettera > lettera2`** [*Invio*]

Converte il file '`lettera`', scritto con la codifica BIG5 (usata per la lingua cinese), a UTF-8, generando il file '`lettera2`'.

47.8.3 Utilizzo di «luit»

«

Il programma '**`luit`**', il quale fa parte di X (capitolo 28), è un filtro che si utilizza per avviare un altro programma, quando il proprio terminale a caratteri è configurato in modo da gestire la codifica UTF-8 (sia per la tastiera, sia per lo schermo) e il programma in questione utilizza una codifica differente:

```
luit [opzioni] [--] [programma [argomenti]]
```

Come si può intuire dal modello sintattico, in mancanza dell'indicazione di un programma da avviare, '**`luit`**' avvia una shell. Al posto di descrivere le opzioni di questo programma, vengono mostrati alcuni esempi, a cominciare da quello più semplice, in cui

il controllo della conversione avviene semplicemente attraverso la configurazione della variabile ***LC_ALL***:

```
$ LC_ALL=en_US.ISO-8859-1 luit mio_programma [Invio]
```

In questo caso si avvia il programma ***mio_programma*** specificando per lui la variabile di ambiente ***LC_ALL*** con il valore che si può vedere. Il programma '**luit**' fa in modo che i dati provenienti dalla tastiera siano convertiti da UTF-8 a ISO 8859-1, facendo l'opposto per i dati diretti dal programma allo schermo. Si osservi, comunque, che la configurazione locale del tipo '**en_US.ISO-8859-1**' deve essere disponibile nel sistema.

```
$ LC_ALL=en_US luit -encoding "ISO 8859-1" mio_programma [Invio]
```

In questo caso, si rende esplicita la codifica con cui deve funzionare il programma '**mio_programma**', attraverso l'opzione '**-encoding**', secondo la notazione prevista da '**luit**'. Per la precisione, si può ottenere l'elenco di tutte le codifiche previste, secondo la notazione di '**luit**', con l'opzione '**-list**':

```
$ luit -list [Invio]
```

Si osservi che quando ci si collega a un elaboratore remoto, nel quale non è prevista una configurazione locale con una codifica UTF-8, è necessario usare '**luit**' come già mostrato, per esempio così:

```
$ LC_ALL=en_US luit -encoding "ISO 8859-1" ssh nodo [Invio]
```

Nell'esempio si può riconoscere l'uso del programma '**ssh**', ovvero di Secure Shell.

Si osservi che negli esempi è stata usata la variabile di ambiente ***LC_ALL***, perché questa prevale su tutte le variabili ***LC_**** e anche su ***LANG***.

47.9 Analisi lessicale

«

Gli errori che si possono fare scrivendo un testo sono di vario tipo, ma quelli puramente lessicali, ovvero ciò che potrebbe essere classificato come errore di battitura, rappresentano i meno importanti. Tuttavia, si tratta pur sempre di una buona percentuale nell'insieme globale di errori che può contenere un testo.

Un programma banale che sia in grado di mostrare le parole che risultano semplicemente sconosciute, è già un buon aiuto verso l'obiettivo dello scrivere in modo corretto.

Un programma di analisi lessicale è utile quando si può gestire un dizionario personale, perché non si possono escludere le eccezioni da un testo, come per esempio il nome o il cognome di una persona, un indirizzo o una sigla particolare. In presenza di documenti di grandi dimensioni, diventa necessario gestire un dizionario specifico per ognuno di questi, in modo da non interferire con l'analisi di altri in cui certi termini, ammissibili da una parte, non possono esistere dall'altra.

47.9.1 Ispell

«

Ispell[20] è un programma di scansione lessicale che permette la realizzazione di dizionari contenenti anche indicazioni sulle possibili aggregazioni di parole (si pensi alla lingua tedesca in cui le parole sono generate spesso dall'unione di altre).

Lo studio di questa caratteristica di Ispell riguarda chi vuole realizzare un dizionario standard per un linguaggio particolare: generico o specifico di un certo settore. Qui si intende mostrare un uso semplificato di questo programma, in cui si utilizzano dizionari standard e si generano i propri dizionari personali specifici per ciò che si fa.

47.9.1.1 Dizionari

Generalmente, il pacchetto di distribuzione di Ispell contiene un dizionario predefinito (di solito per la lingua inglese). Dovrebbe trattarsi del file '`/usr/lib/ispell/default.hash`' (che comunque di solito è un collegamento simbolico a un altro file). Nella stessa directory vanno collocati altri file per altre lingue, o per linguaggi specifici. Questi file, terminanti con l'estensione '`.hash`', sono ottenuti a partire da una coppia di file di testo, attraverso la compilazione con '**`buildhash`**', ogni volta che si cambia piattaforma.

Il dizionario italiano, in forma sorgente, si compone di due file sorgenti: '`italian.aff`' e '`italian.words`'. Il primo dei due contiene la tabella *affix*, la quale in pratica rappresenta una serie di regole sull'insieme dei caratteri ammissibili e sulla possibile unione di parti di parole, mentre il secondo è l'elenco di parole vero e proprio. Queste parole elencate, contengono a volte dei riferimenti aggiuntivi indicati dopo una barra obliqua ('**`/`**') che hanno valore in base alle definizioni della tabella *affix*. L'approfondimento sulla sintassi del file *affix* è utile solo se si vuole realizzare un dizionario hash specifico, mentre l'utilizzatore normale può ignorare questo problema. La compilazione dei file sorgenti in modo da ottenere un dizionario hash si ottiene con il comando seguente:

```
$ buildhash italian.words italian.aff italian.hash[Invio]
```

Si ottiene il file ‘`italian.hash`’, da collocare nella directory ‘`/usr/lib/ispell/`’. Se si intende utilizzare sistematicamente questo dizionario, si può predisporre la variabile di ambiente ***DICTIONARY***, assegnandovi il nome del file: ‘`italian.hash`’. In alternativa, si può usare ‘**`ispell`**’ con l’opzione ‘**`-d`**’, come nell’esempio seguente (l’estensione ‘`.hash`’ è predefinita e può essere omessa).

```
$ ispell -d italian documento.txt[Invio]
```

I dizionari personali sono invece una cosa diversa: si tratta di un elenco di termini, scritto con le stesse modalità di un sorgente, senza un file *affix* a fianco (o meglio, utilizzando quello del dizionario hash a cui si fa riferimento). Normalmente, tali file personali sono aggiornati da Ispell, quando questo viene usato in modo interattivo. Il nome predefinito del dizionario personale è ‘`~/.ispell_`***linguaggio***’. Per esempio, se si utilizza il dizionario standard predefinito, viene generato e utilizzato il file ‘`~/.ispell_default`’ (nella directory personale), a meno di specificare un nome diverso con le opzioni.

Si osservi che in aggiunta ai file personali ci possono essere dei file più specifici, legati alla directory corrente: ‘`./.ispell_`***linguaggio***’.

47.9.1.2 Avvio e opzioni fondamentali

«

Il modello seguente rappresenta una semplificazione estrema della sintassi dell’eseguibile ‘**`ispell`**’, però, prima di apprendere il funzionamento delle particolarità di questo programma, è meglio comprendere le sue possibilità fondamentali:

```
ispell [opzioni] file_da_analizzare
```

Ispell può funzionare in modo interattivo, oppure no. In teoria, è possibile anche realizzare un programma che sfrutti le funzionalità di Ispell attraverso un condotto; in pratica, si tratta in questo caso dell'utilizzo meno importante che si può fare di Ispell.

Opzione	Descrizione
`-d` ***dizionario_hash***	Permette di specificare un file dizionario differente da quello predefinito (che di solito è '`english.hash`'). Il nome del file viene indicato generalmente senza estensione e senza percorso, facendo implicitamente riferimento alla directory '`/usr/lib/ispell/`' e a file con estensione '`.hash`'.
`-p` ***dizionario_personale***	Permette di specificare un dizionario personale differente da quello predefinito (che di solito è '`~/.ispell_`…').
`-W` ***n_caratteri***	Specifica la lunghezza delle parole che non devono essere prese in considerazione. In pratica, da quel numero di caratteri in giù, si considerano tutte valide.
`-x`	Evita la creazione di una copia di sicurezza. Senza indicare questa opzione, dovrebbe essere salvata una copia del file originale aggiungendo al suo nome l'estensione '`.bak`'.
`-b`	Si tratta dell'opzione opposta a '**`-x`**', in quanto permette di forzare la richiesta di creazione di una copia di sicurezza.

Opzione	Descrizione
`-t`	Fa in modo che il testo da analizzare sia considerato un sorgente TeX, o LaTeX, per il quale si devono ignorare i codici di composizione e possibilmente anche alcune indicazioni che sono solo funzionali a TeX, dal momento che non riguardano il contenuto del testo. Questa dovrebbe essere la modalità predefinita di funzionamento. In generale, questa modalità va bene anche per il testo puro e semplice, purché non ci siano barre oblique inverse che possano essere confuse con comandi di TeX.
`-n`	Fa in modo che il testo da analizzare sia considerato un sorgente Nroff o Troff, per il quale si devono ignorare i codici di composizione. La possibilità di distinguere i codici di composizione di TeX, *roff, o altro, dipende anche dal file *affix* del dizionario utilizzato.

47.9.1.3 Funzionamento interattivo

«

Il funzionamento normale di Ispell è interattivo. Generalmente viene fatta una copia di sicurezza del file analizzato, con un nome che termina con l'aggiunta dell'estensione '`.bak`', quindi Ispell permette di modificare il contenuto del file originale, in base alle scelte dell'utente.

Figura 47.62. Funzionamento interattivo di Ispell.

```
    stai               File: lettera

Ciao come stai?

00: stab     09: st-AI
01: stag
02: staid
03: stain
04: stair
05: Stan
06: star
07: stay
08: st AI

[SP] <number> R)epl A)ccept I)nsert L)ookup U)ncap Q)uit
e(X)it or ? for help
```

La figura 47.62 mostra il caso di un file, denominato '`lettera`', contenente una frase normalissima, in cui la parola «stai» non viene riconosciuta. In effetti, si suppone di avere utilizzato il dizionario hash predefinito, ovvero quello inglese.

La parola '**`stai`**' viene evidenziata se le caratteristiche del terminale lo consentono; in ogni caso, viene indicata a parte, all'inizio (come si vede dall'esempio). Se possibile, Ispell elenca una serie di alternative possibili, in base alle affinità che può avere il termine sconosciuto con altre parole contenute nel dizionario. Questo elenco è numerato, in modo da permetterne la selezione. Nella parte bassa dello schermo appare un menù riepilogativo degli altri comandi a disposizione; comandi che si richiamano prevalentemente con la semplice pressione di tasti o combinazioni di tasti mnemonici.

Comando	Descrizione
[*Spazio*]	Fa in modo che Ispell accetti la parola temporaneamente. Se successivamente Ispell ne trova ancora, queste vengono segnalate di nuovo.
[*R*] [*r*]	Richiede la sostituzione della parola errata con un'altra che deve essere inserita subito dopo. Se anche la nuova parola non sembra valida, questa viene segnalata ugualmente da Ispell. La sostituzione riguarda solo quell'occorrenza particolare; se viene ritrovato ancora lo stesso errore, Ispell continua a segnalarlo.
[*A*] [*a*]	Fa sì che Ispell ignori la parola per tutto il resto del documento.
[*I*] [*i*]	Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, esattamente com'è, rispettando maiuscole e minuscole.
[*U*] [*u*]	Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, senza distinguere tra maiuscole e minuscole.
[*0*] [*1*] ... [*0*][*0*] [*0*][*1*] ...	La selezione di un numero fa riferimento alle voci proposte come parole alternative a quella errata. Con questa selezione di intende ottenere la sostituzione delle parole. È importante osservare che, se l'elenco supera le nove unità, la selezione avviene con due cifre numeriche. L'esempio che appare nella figura mostra questo caso: per indicare la parola '**stag**', occorre la sequenza [*0*][*1*].

Comando	Descrizione
[*X*] [*x*]	Conclude il lavoro completando la scrittura del file e ignorando altri errori eventuali. Chiude anche il file del dizionario personale, mantenendo le voci aggiunte fino a quel punto.
[*Q*] [*q*]	Termina immediatamente, lasciando inalterato il file, senza conservare i termini eventualmente annotati per l'aggiunta nel dizionario personale.
[*Ctrl l*] [*Ctrl L*]	Ripulisce lo schermo.

Per quanto riguarda il funzionamento interattivo di Ispell, sono importanti due opzioni.

Opzione	Descrizione
`-M`	Richiede espressamente la visualizzazione del menù riassuntivo dei comandi interattivi. Di solito, tale menù appare in modo predefinito, a meno di avere compilato Ispell con opzioni particolari.
`-N`	Fa in modo che il menù riepilogativo dei comandi non venga visualizzato.

Segue la descrizione di alcuni comandi.

- `$` **`ispell -d italian lettera`**[*Invio*]

  Analizza il file '`lettera`' utilizzando il dizionario hash '**`italian`**', ovvero, il file '`/usr/lib/ispell/italian.hash`', e il dizionario personale predefinito: '`~/.ispell_italian`'.

- `$` **`ispell -d italian -p mio lettera`**[*Invio*]

  Come nell'esempio precedente, ma in questo caso si utilizza il dizionario personale rappresentato dal file '`./mio`'.

47.9.1.4 Funzionamento non interattivo

«

Quando Ispell funziona in modo non interattivo, si limita a generare un elenco di termini, anche ripetuti, che risultano sconosciuti in base al dizionario. Ispell può anche essere utilizzato attraverso un altro programma, quando si indica l'opzione '**`-a`**', ma si tratta di un modo un po' complicato che qui non viene descritto.

Per ottenere l'elenco dei termini sconosciuti, si utilizza l'opzione '**`-l`**'. Per esempio, questa possibilità di Ispell può essere sfruttata per produrre rapidamente un dizionario personale.

Se si dispone di un testo della cui esattezza si è certi, si può ottenere da Ispell l'elenco dei termini da lui sconosciuti, generando poi un dizionario personale con tutte queste eccezioni. Si procede nel modo seguente:

`$` **`ispell -d italian -l < romanzo > mio_dizionario`**[*Invio*]

In questo modo, tutti i termini contenuti nel file '`./romanzo`' che non risultano dal dizionario hash '**`italian`**', vengono emessi attraverso lo standard output e diretti nel file '`./mio_dizionario`'.

`$` **`sort -f < mio_dizionario > dizionario1`**[*Invio*]

In questo modo si riordina l'elenco di parole ottenuto, generando il file '`./dizionario1`', dove l'opzione '**`-f`**' serve a non distinguere tra lettere minuscole e maiuscole, anche se restano i doppioni. Con

questo elenco si vuole generare un dizionario personale, eliminando questi doppioni ed eventualmente generando altre semplificazioni.

```
$ munchlist -s italian -l italian.aff dizionario1 ↩
↪> dizionario2 [Invio]
```

In questo modo, si ottiene il compattamento del file '`./dizionario1`', in base a quanto già contenuto del dizionario hash '**`italian`**' e secondo le regole del file *affix* '`./italian.aff`', generando il file '`./dizionario2`', che finalmente può essere utilizzato come dizionario personale.

In alternativa, si può anche tentare di dare in pasto a Ispell il file ottenuto dopo l'ordinamento, senza filtrarlo attraverso '**`munchlist`**'; lasciando che sia Ispell stesso a eliminare i doppioni.

47.9.1.5 Programmi di servizio di contorno a Ispell

Ispell si compone di diversi file binari. Il più importante è '**`ispell`**', come si è visto, ma altri sono necessari per la gestione dei file di dizionario. Si è già accennato a '**`buildhash`**' e a '**`munchlist`**', il cui utilizzo è il caso di riepilogare.

```
buildhash dizionario_sorgente file_affix dizionario_hash
```

```
munchlist [-l file_affix] [-s dizionario_hash] [elenco_da_ridurre] ↩
↪> elenco_ridotto
```

Quelle mostrate sono le sintassi semplificate di questi due programmi. Di più può essere appreso dalla lettura di *ispell(1)*. Segue la descrizione di alcuni esempi.

- `$` **`munchlist mio_dizionario > dizionario`**[*Invio*]

  Utilizza il dizionario hash e il file *affix* standard per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- `$` **`munchlist -s italian -l ./italian.aff mio_dizionario`** ↩
  ↪**`> dizionario`**[*Invio*]

  Utilizza il dizionario hash '**`italian`**' ('`/usr/lib/ispell/italian.hash`') e il file *affix* '`./italian.aff`' per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- `$` **`buildhash italian.words italian.aff italian.hash`**[*Invio*]

  Genera il dizionario hash '`./italian.hash`', a partire dall'elenco '`./italian.words`' e dal file *affix* '`./italian.aff`'.

47.9.1.6 Gestione dei dizionari personali

«

L'utilizzo occasionale di Ispell richiede la presenza di un dizionario hash e probabilmente di uno personale predefinito, che quasi sicuramente è '`~/.ispell_italian`'. Ma la correzione ortografica basata esclusivamente su un dizionario è tanto più efficace quanto minore è il numero delle parole previste, ovvero, quanto più specifico è il dizionario utilizzato.

Di fronte alla realizzazione di un documento di un certo impegno, o di una serie di documenti che trattano dello stesso genere di cose, potrebbe essere conveniente utilizzare un dizionario personale specifico per quel progetto, eventualmente partendo da un dizionario hash praticamente vuoto.[21]

Per realizzare un dizionario «vuoto», adatto a qualunque linguaggio che utilizzi la codifica ISO 8859-1, si potrebbe partire dal file *affix* che contiene solo le righe seguenti, il cui unico scopo è quello di ammettere l'uso di tutte le lettere accentate e speciali.[22]

```
# minimo.aff
# Accetta qualunque carattere accentato e speciale di
# ISO 8859-1.

wordchars       [a-z]   [A-Z]
wordchars       [à-\376]        [À-\336]
wordchars       [\337]
wordchars       [\377]

prefixes

suffixes
```

Le parole chiave '**prefixes**' e '**suffixes**' sono obbligatorie, ma il file è ancora incompleto (viene segnalato dai programmi come '**buildhash**' e '**munchlist**'), anche se funziona ugualmente per lo scopo che ci si prefigge qui.

Volendo esagerare, se le cifre numeriche possono avere un ruolo nella composizione delle parole che si vogliono controllare, si può aggiungere anche la riga seguente, tenendo conto che però poi '**munchlist**' non funziona tanto bene.[23]

```
wordchars       [0-9]
```

A fianco di questo si deve creare un elenco di parole che ne contenga almeno una, come nell'esempio seguente:

```
Linux
```

Si suppone che il file *affix* sia stato nominato ‘`minimo.aff`’ e che l’elenco sia ‘`minimo.sml`’. Per creare il file hash, si procede come è già stato presentato più volte.

```
$ buildhash minimo.sml minimo.aff minimo.hash [Invio]
```

Pur con una segnalazione di errore, dovuta all’estrema semplicità del file *affix*, si ottiene il file ‘`minimo.hash`’ nella directory corrente. Questo file hash può essere usato solo per testi normali, senza codici di composizione di alcun tipo, dal momento che il file *affix* mostrato non è stato predisposto per questo.

Se si dispone di un documento ritenuto sicuro, si può generare il dizionario personale relativo.

```
$ ispell -d ./minimo.hash -l < documento.txt > elenco [Invio]
```

In questo modo si ottiene l’elenco delle parole usate nel file ‘`documento.txt`’, le quali sono praticamente tutte sconosciute. Questo elenco deve essere riordinato e ridotto.

```
$ sort -f < elenco > elenco1 [Invio]
```

```
$ munchlist -l minimo.aff -s minimo.hash elenco1 ↩
↪> dizionario [Invio]
```

Dopo la riduzione si ottiene finalmente il dizionario personale specifico del documento; successivamente si possono eseguire le verifiche sullo stesso documento di origine (a seguito di aggiunte o di modifiche), con il comando seguente:

```
$ ispell -d ./minimo.hash -p ./dizionario documento.txt [Invio]
```

47.9.2 Aspell

Aspell[24] è un programma funzionalmente simile a Ispell, che gestisce dei dizionari propri ed è in grado di utilizzare quelli appropriati alla configurazione locale:

```
aspell [opzioni] comando
```

I dizionari di Aspell dovrebbero trovarsi nella directory '`/usr/lib/aspell/`', assieme a dei file che gli consentono di riconoscere le codifiche più comuni dei caratteri. Vengono mostrati solo alcuni esempi di funzionamento del programma, senza descrivere le opzioni; per questo si può consultare la pagina di manuale *aspell(1)*, oppure la documentazione Info: *info aspell*.

- `$` **`aspell -c lettera`**[*Invio*]

  Esamina il file '`lettera`', utilizzando il dizionario predefinito per la configurazione locale esistente. Se il programma incontra parole sconosciute, consente di intervenire in modo interattivo, come nella schermata seguente:

```
Ciao Antonio,
come stai?
È moltoa che non ci si sente; cosa fai di bello?
Daniele
---------------------------------------------------------
1) molta                        2) molto



i) Ignora                       I) Ignora tutti
r) Rimpiazza                    R) Rimpiazza tutti
a) Aggiungi                     l) Aggiungi minuscolo
b) Abbandona                    x) Esci
---------------------------------------------------------
?
```

Come si vede, in questo caso la configurazione è quella italiana, dato che anche il menù appare tradotto. Il testo da verificare appare con qualche riga in più rispetto a Ispell, facilitando così il lavoro di correzione.

- `$` **`aspell --lang=it -c lettera`**[*Invio*]

  Esamina il file '`lettera`', utilizzando espressamente il dizionario italiano.

- `$` **`aspell --encoding=utf-8 --lang=it -c lettera`**[*Invio*]

  Esamina il file '`lettera`', utilizzando espressamente il dizionario italiano e la codifica UTF-8.

47.9.3 Myspell e Hunspell

«

Myspell è un componente di OpenOffice.org 1.∗ e di altri programmi, per la gestione dei dizionari ortografici. Hunspell[25] è un compo-

nente simile, nato per OpenOffice.org 2.*, con la capacità di gestire meglio il riconoscimento delle parole composte.

In generale, comunque, si tratta sempre di lavori derivati o comunque ispirati al funzionamento di Ispell. In particolare, i dizionari realizzati per Myspell, sono utilizzabili anche da Hunspell.

Generalmente è possibile usare Hunspell anche attraverso un programma frontale simile a Ispell. Supponendo che i dizionari Myspell siano installati a partire da '`/usr/share/myspell/dicts/`', si potrebbe usare Hunspell come si vede nell'esempio seguente:

```
$ hunspell -d /usr/share/myspell/dicts/it-IT lettera[Invio]
```

Così facendo, il programma utilizza il dizionario composto dai file '`/usr/share/myspell/dicts/it-IT.aff`' e '`/usr/share/myspell/dicts/it-IT.dic`', per esaminare il file '`lettera`'.

47.10 Analisi sintattica e stilistica con Textchk

L'analisi sintattica di un testo è un problema ben più complicato della semplice verifica delle parole con un dizionario. Esistono però alcuni tipi di errori sintattici, o stilistici, che si possono identificare con l'aiuto di espressioni regolari (*regular expression*).

La lingua italiana consente spesso l'utilizzo di forme espressive differenti, per le quali dovrebbe esserci almeno uniformità all'interno di uno stesso documento. Per esempio, occorre decidere se si vuole scrivere: «una aula» oppure «un'aula», «ed anche» oppure «e anche»,...

In questa sezione si mostra un programma Perl che può aiutare a definire delle regole rappresentate in forma di espressioni regolari, per segnalare degli errori sintattici o stilistici. Con questo programma è

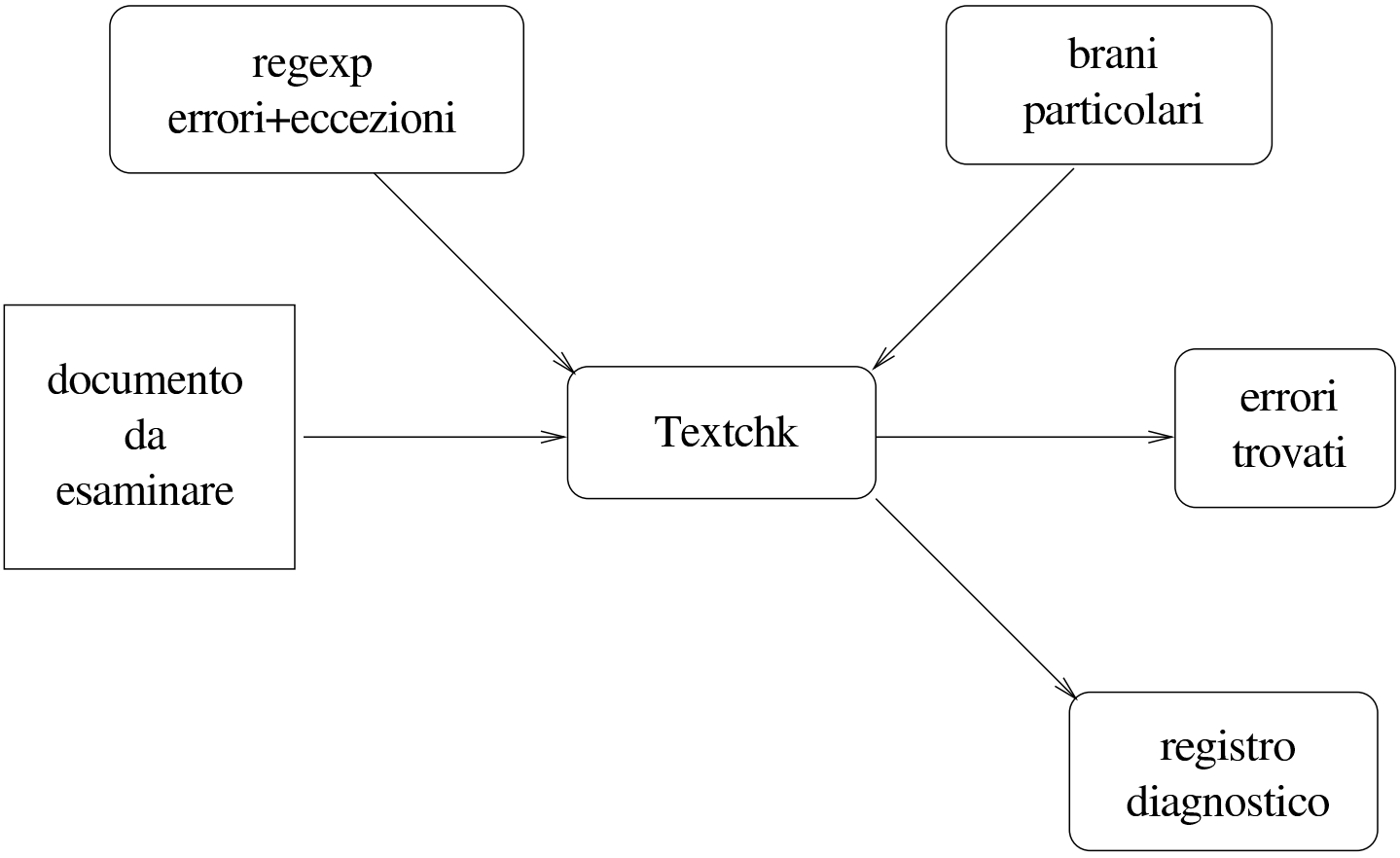
possibile indicare anche delle regole di eccezione e delle particolarità riferite a un solo documento. Il programma in questione è Textchk: *allegati/textchk/textchk.pl* [26]

47.10.1 Principio di funzionamento

«

Textchk scandisce un file di partenza generando un altro file contenente le parti di testo che risulterebbero errate (oltre a un file diagnostico contenente la registrazione del procedimento di verifica). Prima di iniziare a leggere il file da esaminare, vengono caricati dei modelli che esprimono degli errori, espressi in forma di espressione regolare, seguiti eventualmente da dei modelli di eccezione. Infine, vengono caricate anche delle particolarità riferite al testo che si elabora, trattate in forma letterale e non più secondo il modello di un'espressione regolare.

Figura 47.69. Schema di funzionamento di Textchk.



Gli errori che si possono ricercare attraverso delle espressioni regolari, riguardano la vicinanza di parole che hanno caratteristiche determinate, come l'uso o meno di articoli apostrofati. Sotto questo aspetto, diventa importante che, nel file di testo originale, ogni paragrafo si trovi su una sola riga, cioè non sia interrotto su più righe.

A fianco di questo problema, si aggiunge il fatto che il file sorgente che si vuole esaminare potrebbe contenere dei codici di controllo, come nel caso di TeX (o LaTeX) e di HTML. In tutte queste situazioni, prima di passare all'analisi vera e propria, occorre ripulire e riadattare il testo, in modo da avere a che fare con un file di testo puro, in cui ogni paragrafo si trovi su una sola riga.

47.10.1.1 Espressioni regolari

«

Textchk è scritto in Perl, pertanto le espressioni regolari che possono essere gestite sono quelle di questo linguaggio di programmazione.

La ricerca della corrispondenza con le espressioni regolari che esprimono un errore, viene fatta in modo da circoscrivere, se possibile, tre parole prima e dopo della zona dell'errore. Per questa ragione, non ha senso tentare di identificare l'inizio e la fine di una riga (con i simboli '**^**' e '**$**'), inoltre non è possibile utilizzare le parentesi tonde.

A titolo di esempio, si propone il problema della «d» eufonica, per la precisione il caso di «ad». Supponendo di volerla utilizzare solo quando la parola successiva inizia con la vocale «a», escludendo il caso in cui la parola continui con un'altra «d» (per esempio: «ad amare», ma non «ad adattare»), si possono usare le espressioni regolari seguenti per individuare gli errori.

```
\ba[[:space:]]+a[^d][[:word:]]*\b
\bad[[:space:]]+ad[[:word:]]*\b
\bad[[:space:]]+[^a][[:word:]]*\b
```

Per intendere meglio il significato di ciò che è scritto, la prima riga significa:

'**\b**'	lo spazio vuoto prima della parola;
'**a**'	la lettera «a»;
'**[[:space:]]+**'	uno o più spazi orizzontali;
'**a[^d]**'	la lettera «a» seguita immediatamente da qualunque cosa che sia diversa dalla lettera «d»;
'**[[:word:]]***'	zero o più caratteri alfabetici;
'**\b**'	lo spazio vuoto dopo la parola.

Nello stesso tempo, però, si può decidere di accettare un'eccezio-

ne: «ad esempio», che secondo quanto stabilito con l'ultima delle espressioni regolari appena mostrate, dovrebbe essere un errore. Si può usare quindi l'espressione regolare seguente, tra le eccezioni.

```
\bad[[:space:]]+esempio\b
```

47.10.2 Configurazione

La configurazione di Textchk serve a definire gli errori sintattici che si ricercano. In generale è importante definire una configurazione specifica per ogni singolo progetto di documentazione, ma resta la possibilità di stabilire regole personali, legate all'utente, oltre che regole generali legate al sistema (per quanto questo possa avere un valore relativo).

La configurazione avviene attraverso un file di testo normale, in cui le righe bianche, quelle vuote e quelle che iniziano con il simbolo '**#**' vengono ignorate. Le altre righe sono dei record che possono avere una delle due forme seguenti:

```
DBL_____regola_di_errore [_____testo_esplicativo]
```

```
ERR_____regola_di_errore [_____testo_esplicativo]
```

```
EXC_____regola_di_eccezione
```

Nel primo caso si identifica una parola che si ritiene possa essere stata scritta due volte, in modo erroneo; il secondo indica un modello di errore, mentre nel terzo si tratta di un'eccezione. I record che descrivono le regole di eccezione si riferiscono sempre all'ultima re-

gola di errore (di tipo '**DBL**' o '**ERR**') che sia stata incontrata fino a quel punto.

La forma di questi record è un po' strana, nel senso che la separazione dei campi avviene attraverso una sequenza di quattro trattini bassi ('____'). Ciò serve per evitare di creare problemi alla realizzazione delle espressioni regolari che descrivono gli errori e le eccezioni.

```
#
# d eufonica
# a|e|o prendono una «d» eufonica se sono seguite da una
# parola che inizia con la stessa vocale, a meno che ci sia
# subito dopo un'altra «d».
#
ERR____\ba[[:space:]]+a[^d][[:word:]]*\b____a --> ad
EXC____\bda[[:space:]]+a[[:space:]]+a\b
#
ERR____\bad[[:space:]]+ad[[:word:]]*\b____ad --> a
#
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
#
ERR____\be[[:space:]]+e[^d][[:word:]]*\b____e --> ed
ERR____\bed[[:space:]]+[eE]d[[:word:]]*\b____ed --> e
ERR____\bed[[:space:]]+[^eèE][[:word:]]*\b____ed --> e
#
ERR____\bo[[:space:]]+[oO][^d][[:word:]]*\b____o --> od
ERR____\bod[[:space:]]+[oO]d[[:word:]]*\b____od --> o
ERR____\bod[[:space:]]+[^oO][[:word:]]*\b____od --> o
```

L'esempio mostra una serie di istruzioni con le quali si cerca di definire l'uso della «d» eufonica. Vale la pena di analizzare cosa succede di fronte a una situazione precisa. Si suppone di avere scritto un te-

sto nel quale è stata inserita la frase seguente, la quale risulta essere disposta su una sola riga:

```
Purtroppo, fino ad ora il colore dell'auto non è stato ↩
↪scelto dal cliente.
```

Concentrando l'attenzione sui record di configurazione seguenti, si può simulare ciò che succede.

```
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
```

Per cominciare, viene individuato un errore in via preliminare in corrispondenza di «ad ora», perché la parola che segue «ad» non inizia con una lettera «a». Textchk preleva una stringa di tre parole prima e tre parole dopo questo errore: «Purtroppo, fino ad ora il colore dell'auto». In questo caso, le parole precedenti sono solo due, perché non è stato possibile ottenere di più.

Su questa stringa estratta viene condotto il controllo per le eccezioni successive; così, dal momento che si ottiene una corrispondenza (sempre con «ad ora»), l'errore si rivela infondato (in base ai presupposti stabiliti).

L'ultimo campo dei record che descrivono gli errori serve per indicare una spiegazione per ciò che viene identificato come un errore. Questa spiegazione viene mostrata da Textchk nel momento in cui l'errore relativo viene mostrato, secondo lo schema seguente:

testo_esplicativo
    tre_parole_precedenti*>>*errore*<<*tre_parole_successive

Come si vede, la corrispondenza con l'errore viene evidenziata dai

delimitatori ‘>>’ e ‘<<’.

47.10.2.1 Gerarchia della configurazione

«

Textchk è stato pensato originariamente per avere una configurazione specifica per ogni progetto di documentazione che ogni autore possa gestire. Tuttavia, è possibile definire anche una configurazione personale e una di sistema. Si tratta dei file seguenti:

‘./.textchk.rules’	contiene la configurazione corrente, che viene letta prima delle altre;
‘~/.textchk.rules’	contiene la configurazione personale, letta subito dopo quella corrente;
‘/etc/textchk.rules’	contiene la configurazione di sistema, che viene letta alla fine.

In generale non è opportuno stabilire una configurazione generale di sistema. Tuttavia, se c’è la necessità di annullare l’effetto di una regola di errore stabilita a livello generale, si può dichiarare la stessa regola nella configurazione personale o in quella corrente, facendola seguire immediatamente da un’eccezione identica. Per esempio, si può supporre di avere definito a livello di sistema la regola seguente, che richiede l’uso della «d» eufonica ogni volta che la parola seguente inizia con una vocale:

```
ERR____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b____a/e/o --> ad/ed/od
```

Per annullarne l’effetto completamente, basta aggiungere la stessa regola in qualità di eccezione, subito dopo:

```
#
# Regola di sistema che qui viene annullata.
#
ERR____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b____a/e/o --> ad/ed/od
EXC____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b
```

47.10.2.2 Casi particolari

Alle volte non conviene indicare troppe eccezioni, oppure non è materialmente possibile. Per esempio, si può immaginare il caso in cui si vuole mostrare veramente un modo sbagliato di scrivere per qualche ragione. Per queste situazioni viene in aiuto un file di configurazione aggiuntivo, che però può essere associato esclusivamente a un solo progetto di documentazione. Si tratta del file '`./.textchk.special`', in cui si possono inserire integralmente alcune stringhe che Textchk ha indicato precedentemente come errate.

Per questa parte della configurazione non c'è molto da fare: basta utilizzare un programma per la creazione e la modifica dei file di testo ricopiando ciò che serve dal file che viene generato da Textchk per registrare gli errori trovati. L'esempio seguente mostra un estratto di quello che potrebbe contenere questo file. Si osservi il fatto che si tratta di esempi di errori scritti così di proposito.

```
oppure «un'aula», «ed anche» oppure «e
vuole scrivere: «una aula» oppure «un'aula»,
ma non «ad adattare»), si possono
```

47.10.2.3 L'indicazione di parole doppie

Un errore frequente nella scrittura di un testo consiste nella ripetizione di una parola per due volte di seguito, mentre l'intenzione sarebbe quella di scriverla una volta sola. Per intercettare questo tipo di situazione si utilizza il record '**`DBL`**'. Nel campo dell'espressione che

indica l'errore, si fa riferimento implicitamente a una parola intera. Infatti, nella comparazione reale, viene aggiunto il simbolo '**\b**' all'inizio e alla fine, a sottolineare che la parola deve essere completa. Si osservi l'esempio seguente:

```
#
# Parole doppie.
#
DBL____[[:word:]][[:word:]]+____Due parole identiche
EXC____\bciao[[:space:]]+ciao\b
```

L'intenzione è di individuare qualunque parola ('**[[:word:]]+**'), composta almeno da due caratteri, che si ripete immediatamente. Viene posta una sola eccezione alla coppia «ciao ciao».

47.10.3 Come si usa

Textchk si compone di un eseguibile unico, '**textchk**', che si utilizza secondo lo schema sintattico seguente:

```
textchk --input-type=tipo_di_file  file_da_analizzare ↩
↪       [errori_risultanti  [file_diagnostico ] ]
```

```
textchk --help
```

```
textchk --version
```

Oltre alle opzioni standard, '**--help**' e '**--version**', l'opzione '**--input-type**' serve a stabilire il tipo di file che si fornisce in ingresso, in modo che Textchk sappia come fare per gestirlo opportunamente, attraverso un argomento:

Opzione	Descrizione
`--input-type=standard`	si riferisce a un file di testo in cui ogni capoverso occupa esattamente una riga e non richiede altri adattamenti;
`--input-type=man`	si riferisce a un file Troff delle pagine di manuale;
`--input-type=texinfo` `--input-type=texi`	si riferisce a un sorgente Texinfo;
`--input-type=html`	si riferisce a un file HTML.

Il secondo argomento della riga di comando è il nome del file da analizzare, che deve corrispondere al tipo indicato precedentemente. Il terzo argomento serve a definire il nome del file che viene creato per annotare le stringhe errate che vengono individuate; se non viene fornito espressamente il suo nome, viene creato un file con lo stesso nome di quello in ingresso, con l'aggiunta dell'estensione '`.tchk`' ('***file_da_analizzare*** `.tchk`'). Il quarto argomento serve a specificare il nome del file diagnostico, nel quale vengono registrate tutte le fasi di individuazione di errori e di eccezioni. Anche l'indicazione di questo file può essere omessa; in tal caso viene usato il nome del file degli errori con l'aggiunta dell'estensione '`.tdiag`', oppure il file in ingresso con la stessa aggiunta ('***errori_risultanti*** `.tdiag`' oppure '***file_da_analizzare*** `.tdiag`').

Per esempio, il comando seguente genera i file '`bash.1.tchk`' e '`bash.1.tdiag`':

```
$ textchk --input-type=man bash.1 [Invio]
```

47.10.3.1 Come vengono mostrati gli errori e i dati diagnostici

«

Durante il suo lavoro, Textchk mostra sullo schermo ciò che trova, delimitando gli errori tra i delimitatori '**>>**' e '**<<**'. Per esempio, in base alle regole seguenti,

```
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
```

si possono ottenere segnalazioni come queste:

```
ad --> a
    Pertanto, andando >>ad elevare<< il proprio livello
ad --> a
    contrario, riuscendo così >>ad esplorare<< il proprio
```

Nel file che elenca gli errori si trovano le righe seguenti:

```
Pertanto, andando ad elevare il proprio livello
contrario, riuscendo così ad esplorare il proprio mondo
```

Inoltre, nel file diagnostico si trova l'intero procedimento:

```
??? Pertanto, andando >>ad elevare<< il proprio livello
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
!!! Pertanto, andando >>ad elevare<< il proprio livello

??? contrario, riuscendo così >>ad esplorare<< il proprio mondo
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
!!! contrario, riuscendo così >>ad esplorare<< il proprio mondo

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
EXC \bad[[:space:]]+esempio\b

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
```

```
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
EXC \bad[[:space:]]+esempio\b

??? Pertanto, l'espressione «>>ad emettere<<» non è corretta.
ERR \bad[[:space:]]+[^aA]][[:word:]]*\b
SPC Pertanto, l'espressione «ad emettere» non è corretta.
```

Il file diagnostico mostra informazioni diverse, distinte attraverso una sigla iniziale. Le righe che iniziano con '**???**' indicano il problema trovato; le righe che iniziano con '**ERR**' rappresentano la regola di errore in base alla quale viene evidenziato il problema; le righe che iniziano con '**EXC**' indicano una regola di eccezione per la quale il problema viene superato; le righe che iniziano con '**SPC**' rappresentano un caso particolare (speciale), per cui la frase in questione viene accettata così come si trova. Infine, le righe che iniziano con '**!!!**' rappresentano la conferma finale che si deve trattare di un errore.

47.10.4 Come si installa

Textchk si compone di un solo programma Perl: '**textchk**'. Questo file può essere collocato ovunque sia ritenuto più conveniente, preferendo evidentemente una directory elencata all'interno della variabile di ambiente ***PATH***.

Trattandosi di un programma Perl, deve essere disponibile l'interprete relativo. Attualmente si prevede che questo corrisponda esattamente all'eseguibile '`/usr/bin/perl`'. Se il proprio sistema non è organizzato in questo modo, basta modificare la prima parte del programma:

```
#!/usr/bin/perl
...
```

Dopo la soluzione di questo problema, c'è solo bisogno di predisporre un file di regole, '`./.textchk.rules`', poi, mano a mano che il lavoro procede, può essere conveniente predisporre anche il file '`./.textchk.special`'.

47.10.4.1 Gettext

«

I messaggi che può mostrare Textchk possono essere tradotti, dal momento che viene usato il modulo Perl-gettext. Nel pacchetto del sorgente è presente un file di messaggi per la lingua italiana, che però deve essere compilato e installato:

```
$ msgfmt -o textchk.mo it.po[Invio]
```

In questo modo, si genera il file '`textchk.mo`', che probabilmente va collocato nella directory '`/usr/share/locale/it/LC_MESSAGES/`'.

47.10.4.2 Dipendenze

«

Per funzionare, Textchk richiede l'interprete Perl e la presenza di un modulo speciale: Perl-gettext. Inoltre, per poter gestire correttamente i diversi tipi di file per cui è stato predisposto, richiede in particolare Groff, Lynx e Texinfo.

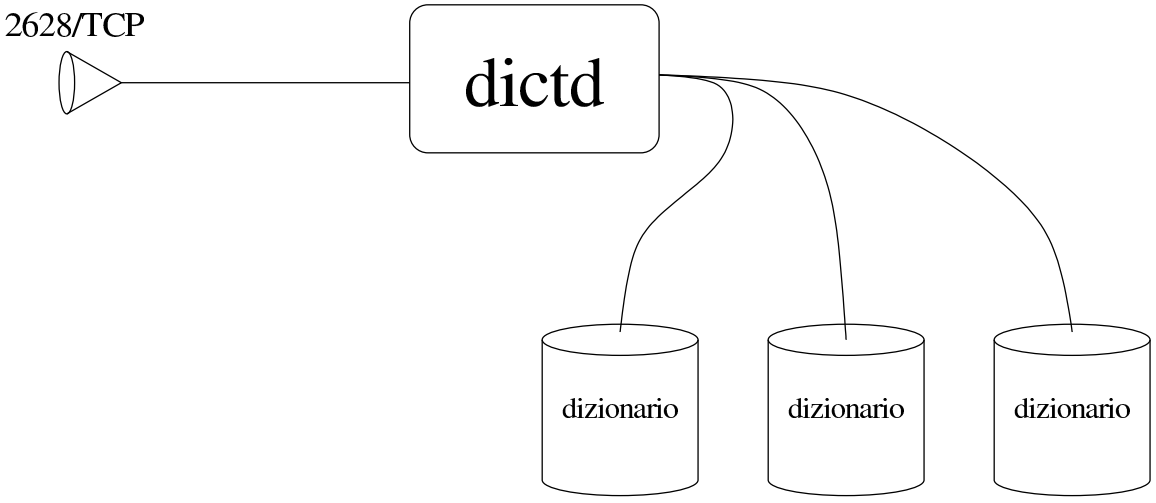
47.11 Dizionari

«

Esistono diversi tipi di dizionario che spesso sono disponibili in forma di file di testo, redatti secondo un certo standard. Non si tratta esclusivamente di dizionari per le traduzioni, ma si possono includere quelli per i sinonimi, quelli specifici per un certo settore tecnico, fino a quelli biblici.

Per accedere a tutti i dizionari disponibili presso un elaboratore, si usa generalmente un programma demone, a cui si accede attraverso la rete, precisamente secondo il protocollo DICT (*Dictionary server protocol*) descritto dal documento RFC 2229. Naturalmente, i file contenenti i dizionari, devono essere realizzati secondo il formato atteso dal demone stesso. Di solito si considera il programma '**`dictd`**' per questo scopo.

Figura 47.87. Il servizio DICT viene fornito attraverso un demone, di solito in ascolto della porta 2628/TCP.



47.11.1 Dictd o Serpento

Il programma che di solito si usa come servente DICT è Dictd[27] oppure Serpento[28] (il secondo sembra avere una gestione migliore dell'insieme di caratteri universale).

```
dictd
```

```
serpento
```

Il programma, sia '**dictd**', sia '**serpento**', salvo indicazioni differenti nella sua configurazione, si collega ai file che dovrebbero trovarsi nella directory '`/usr/share/dictd/`' e si mette in ascolto della porta 2628/TCP.

I file in questione devono essere conformi a quanto atteso dai programmi. Nel caso della distribuzione GNU/Linux Debian, i dizionari sono distribuiti in pacchetti separati, il cui nome inizia per '**dict-**'.

47.11.2 Interrogazione manuale di un servizio DICT

«

Per comprendere l'utilità del servizio DICT, prima di avvicinarsi all'uso dei programmi clienti che possono servirsene, conviene provare a utilizzare direttamente il protocollo, con i suoi comandi principali. Ciò consente anche di verificare il funzionamento del servizio. Per l'interrogazione diretta, si può usare Netcat o Telnet:

```
netcat nodo 2628
```

```
telnet nodo 2628
```

Ecco come si potrebbe usare Telnet per accedere all'elaboratore locale, presso il quale è in funzione Serpento:

```
$ telnet localhost 2628 [Invio]

  Trying 127.0.0.1...
  Connected to 127.0.0.1.
  Escape character is '^]'.
  220 hello <> msg

SHOW SERVER [Invio]
```

```
114 server information follows
serpento
.
250 Command complete
```

QUIT [*Invio*]

```
221 bye bye
Connection closed by foreign host.
```

Prima di vedere i comandi più importanti, è necessario chiarire il significato di alcune definizioni che si rifanno a quanto scritto nel documento RFC 2229.

- Ogni dizionario viene considerato costituire una «base di dati», ma qui non si usa questa dizione che sembra impropria.
- La ricerca di un termine avviene attraverso un confronto, il cui metodo può cambiare in base alla necessità. Per esempio si può cercare una corrispondenza esatta, oppure una corrispondenza fonetica. Il metodo di confronto di una ricerca viene definito «strategia»; qui si fa riferimento al metodo di ricerca.

La tabella successiva mostra la sintassi, semplificata di alcuni comandi.

Sintassi	Descrizione
`DEFINE` ***dizionario parola*** `DEFINE` { `!` | `*` } ***parola***	Richiede di trovare la definizione o la traduzione della parola indicata, cercandola nel dizionario stabilito (si cerca una corrispondenza esatta e completa della parola indicata). Se al posto del dizionari si indica un punto esclamativo, si vuole ottenere il risultato nel primo dizionario annoverato dal servizio; se invece si utilizza un asterisco, si vogliono trovare le corrispondenze con tutti i dizionari disponibili.
`MATCH` ***dizionario metodo_ricerca parola*** `MATCH` { `!` | `*` } ***metodo_ricerca parola***	Richiede di trovare una corrispondenza della parola indicata, attraverso l'indicazione di un metodo di ricerca (o strategia). Se al posto del dizionari si indica un punto esclamativo, si vuole ottenere il risultato nel primo dizionario annoverato dal servizio; se invece si utilizza un asterisco, si vogliono trovare le corrispondenze con tutti i dizionari disponibili.
`MATCH` ***dizionario*** `exact` ***parola***	Richiede di trovare una corrispondenza esatta della parola, praticamente come avviene con il comando '**`DEFINE`**'.

Sintassi	Descrizione
`MATCH` ***dizionario*** `prefix` ***parola***	Richiede di trovare una corrispondenza iniziale della parola.
`MATCH` ***dizionario*** `.` ***parola***	Richiede di trovare una corrispondenza secondo il metodo predefinito del servente, che dovrebbe corrispondere al migliore disponibile.
`SHOW DB` `SHOW DATABASE`	Richiede di mostrare l'elenco dei dizionari disponibili.
`SHOW STRAT` `SHOW STRATEGIES`	Richiede di mostrare l'elenco dei metodi di ricerca disponibili (quelli utilizzabili con il comando '**`MATCH`**'.
`SHOW SERVER` `SHOW SERVER`	Richiede di mostrare il nome del servente.
`QUIT`	Conclude il collegamento.

Segue la descrizione di alcuni esempi, partendo dalle interrogazioni necessarie a conoscere quali sono i dizionari disponibili.

- **`SHOW DB`** [*Invio*]

```
110 57 databases here
…
freedict-iri-eng "freedict-iri-eng"
freedict-eng-lat "freedict-eng-lat"
freedict-deu-ita "freedict-deu-ita"
freedict-deu-por "freedict-deu-por"
.
250 Command complete
```

- **SHOW STRAT** [*Invio*]

```
111 9 strategies here
soundex "Match using SOUNDEX algorithm"
suffix "Match suffixes"
metaphone "metaphone algorithm"
substring "Match substring occurring anywhere in word"
re "POSIX 1003.2 regular expressions"
prefix "Match prefixes"
lev "Match words within Levenshtein distance one"
fnmatch "fnmatch-like (* ? as wildcards)"
exact "Match words exactly"
.
250 Command complete
```

I metodi di ricerca '**exact**' e '**prefix**' sono obbligatori, nel senso che devono essere sempre presenti, secondo quanto stabilito dal documento RFC 2229.

- **DEFINE ! ciao** [*Invio*]

In questo caso si chiede la definizione (la corrispondenza esatta) della parola «ciao» con il primo dizionario disponibile:

```
150 1 here you are
151 "ciao" freedict-eng-cro "freedict-eng-cro"
ciao

ćao
.
250 Command complete
```

In questo caso è stata ottenuta la traduzione dall'inglese al croato.

- **`DEFINE * ciao`**[*Invio*]

  Questa volta si vuole la definizione della parola da tutti i dizionari disponibili; qui si vede solo l'ultimo:

```
150 5 here you are
…
151 "ciao" freedict-ita-eng "freedict-ita-eng"
ciao
     hello
.
250 Command complete
```

- **`MATCH ! soundex amore`**[*Invio*]

  Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**`soundex`**', limitando l'interesse al primo dizionario che contenga almeno una corrispondenza valida:

```
152 9 here you are
…
freedict-fra-eng "amour"
freedict-fra-eng "aumônier"
freedict-fra-eng "annuaire"
freedict-fra-eng "amener"
.
250 Command complete
```

Eventualmente, dopo si può fare una ricerca con il comando '**DEFINE**', specificando anche il dizionario, per avere maggiori indicazioni su una parola dell'elenco trovato.

47.11.3 Il programma Dict per l'interrogazione del servizio

«

Dict[29] è il programma più comune per l'interrogazione di un servizio DICT. Si usa a riga di comando, ma proprio per questo è molto semplice ed efficace:

```
dict [opzioni] parola
```

```
dict [opzioni] dict://nodo/d:parola [:dizionario]
```

```
dict [opzioni] dict://nodo/m:parola↩
↪[:dizionario [:modalità_di_ricerca]]
```

La prima forma di utilizzo del programma implica la ricerca della parola indicata presso il servente o i serventi indicati nella configurazione (salvo indicazione diversa attraverso le opzioni); la seconda richiede una ricerca presso un elaboratore stabilito, utilizzando un confronto completo (l'equivalente del comando '**DEFINE**' del protocollo DICT); l'ultima richiede una ricerca presso un elaboratore stabilito, utilizzando un confronto che non sia necessariamente esatto (l'equivalente del comando '**MATCH**' del protocollo DICT).

Il file di configurazione generale di Dict può essere, a seconda dei sistemi, '`/etc/dict.conf`', oppure '`/etc/dictd/dict.conf`';

il file di configurazione personale di ogni utente è '`~/.dictrc`'. La configurazione serve a specificare i serventi predefiniti, aggiungendo eventualmente indicazioni sulla porta TCP da utilizzare, nel caso non sia quella standard:

```
server nodo [port n_porta]
```

Eventualmente sono disponibili altre opzioni nella direttiva '**`server`**' nei casi in cui l'accesso al servente richieda una forma di autenticazione. È normale trovare queste due direttive nel file di configurazione:

```
server localhost
server dict.org
```

In pratica, si cerca prima un servizio DICT localmente, altrimenti si interroga quello fornito da *dict.org*.

Tabella 47.98. Alcune opzioni per l'utilizzo di Dict.

Opzione	Descrizione
`-h` ***nodo*** `--host` ***nodo***	Specifica di interrogare un servente particolare, che probabilmente è diverso da quanto indicato nella configurazione.
`-p` ***n_porta*** `--port` ***n_porta***	Specifica di rivolgersi a un certo numero di porta TCP, evidentemente diverso da quello predefinito che sarebbe 2628.
`-d` ***dizionario*** `--database` ***dizionario***	Specifica il dizionario all'interno del quale si vuole eseguire la ricerca.

Opzione	Descrizione
`-D` `--dbs`	Richiede un elenco dei dizionari disponibili presso il servente.
`-m` `--match`	Invece di ottenere una «definizione», si richiede di cercare una corrispondenza (sulla base del metodo di ricerca specificato con l'opzione '**`--strategy`**').
`-s` ***metodo_ricerca*** `--strategy` ***metodo_ricerca***	Se si usa l'opzione '**`-m`**', consente di specificare il metodo di ricerca.
`-S` `--strats`	Richiede un elenco dei metodi di ricerca (strategie) disponibili presso il servente.

Segue la descrizione di alcuni esempi.

- `$` **`dict -D`**[*Invio*]

```
Databases available:
…
 freedict-iri-eng freedict-iri-eng
 freedict-eng-lat freedict-eng-lat
 gazetteer2k-places gazetteer2k-places
 freedict-deu-ita freedict-deu-ita
 freedict-por-deu freedict-por-deu
 freedict-deu-por freedict-deu-por
```

- `$` **`dict -S`**[*Invio*]

```
Strategies available:
 soundex    Match using SOUNDEX algorithm
 suffix     Match suffixes
 metaphone  metaphone algorithm
 substring  Match substring occurring anywhere in word
 re         POSIX 1003.2 regular expressions
 prefix     Match prefixes
 lev        Match words within Levenshtein distance one
 fnmatch    fnmatch-like (* ? as wildcards)
 exact      Match words exactly
```

I metodi di ricerca '**exact**' e '**prefix**' sono obbligatori, nel senso che devono essere sempre presenti, secondo quanto stabilito dal documento RFC 2229.

- `$` **`dict \! ciao`**[*Invio*]

  In questo caso si chiede la definizione (la corrispondenza esatta) della parola «ciao» con il primo dizionario disponibile. Il punto esclamativo appare preceduto dalla barra obliqua inversa perché altrimenti la shell lo interpreterebbe con un significato speciale.

```
1 definition found

From freedict-eng-cro [freedict-eng-cro]:

  ciao

  ćao
```

  In questo caso è stata ottenuta la traduzione dall'inglese al croato.

- `$` **`dict -d \* ciao`**[*Invio*]

  Questa volta si vuole la definizione della parola da tutti i dizionari disponibili (anche in questo caso la barra obliqua inversa, che pre-

cede l'asterisco, serve a evitare che la shell interpreti l'asterisco in modo speciale); qui si vede soltanto l'ultimo:

```
6 definitions found
…
From freedict-ita-eng [freedict-ita-eng]:

  ciao
       hello
```

- `$` **`dict ciao`**[*Invio*]

  Si ottiene esattamente la stessa cosa del comando precedente; poiché, se non si specifica il dizionario, vengono scanditi tutti.

- `$` **`dict -m -s soundex amore`**[*Invio*]

  Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**`soundex`**', su tutti i dizionari disponibili:

```
gcide:  Aimer  Ameer  Amir  Amorwe  Amour  Anear  Annoyer
  Annuary  Anoura  anoura  Anura  anuria  Anury  Anywhere
  Aumery
…
foldoc:  annoyware  anr
easton:  Amariah  Aner
hitchcock:  Amariah  Aner
gazetteer:  Amery  Amory  Anmoore  Aynor
gaz-place:  Amery  "Amery, WI"  Amory  Anmoore  Aynor
```

  Eventualmente, dopo si può fare una ricerca normale (come «definizione»), specificando anche il dizionario, per avere maggiori indicazioni su una parola dell'elenco trovato.

- `$` **`dict -D -h dict.org`**[*Invio*]

  Elenca i dizionari disponibili presso l'elaboratore *dict.org*:

```
Databases available:
…
 english    English Monolingual Dictionaries
 trans      Translating Dictionaries
 all        All Dictionaries (English-Only and Translating)
 web1913    Webster's Revised Unabridged Dictionary (1913)
 world95    The CIA World Factbook (1995)
```

- `$` **`dict -S -h dict.org`**[*Invio*]

```
Strategies available:
 exact      Match headwords exactly
 prefix     Match prefixes
 substring  Match substring occurring anywhere in a headword
 suffix     Match suffixes
 re         POSIX 1003.2 (modern) regular expressions
 regexp     Old (basic) regular expressions
 soundex    Match using SOUNDEX algorithm
 lev        Match headwords within Levenshtein distance one
 word       Match separate words within headwords
```

- `$` **`dict dict://dict.org/d:ciao`**[*Invio*]

Si chiede la definizione (la corrispondenza esatta) della parola «ciao» con tutti i dizionari disponibili presso *dict.org*.

```
1 definition found

From WordNet (r) 2.0 [wn]:

  ciao
       n : an acknowledgment that can be used to say hello or
           goodbye (aloha is Hawaiian and ciao is Italian)
           [syn: {aloha}]
```

- `$` **`dict dict://dict.org/d:love:easton`**[*Invio*]

Cerca la definizione della parola «love» nel dizionario '**easton**', presso *dict.org*:

```
1 definition found

From Easton's 1897 Bible Dictionary [easton]:

  Love
     This word seems to require explanation only in the case
     of its use by our Lord in his interview with "Simon,
     the son of Jonas,"
…
       In 1 Cor. 13 the apostle sets forth the excellency of
     love, as the word "charity" there is rendered in the
     Revised Version.
```

- `$` **`dict dict://dict.org/m:amore:\*:soundex`** [*Invio*]

Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**soundex**', su tutti i dizionari disponibili, presso *dict.org*:

```
gcide:  Aimer  Ameer  Amir  Amorwe  Amour  Anear  Annoyer
  Annuary  Anoura  anoura  Anura  anuria  Anury  Anywhere
  Aumery
…
hitchcock:  Amariah  Aner
gazetteer:  Amery  Amory  Anmoore  Aynor
gaz-place:  Amery  "Amery, WI"  Amory  Anmoore  Aynor
```

47.12 Riferimenti

«

- Markus Kuhn, *International standard paper sizes*, *http://www.cl.cam.ac.uk/~mgk25/iso-paper.html*

- R. Smith, F. Wright, T. Hastings, S. Zilles, J. Gyllenskog, *RFC 1759: Printer MIB, Appendix B - Media size names from ISO/IEC 10175 Document printing architecture*, 1995, http://www.ietf.org/rfc/rfc1759.txt
- T. Hastings, R. Herriot, R. deBry, S. Isaacson, P. Powell, *RFC 2911: Internet printing protocol/1.1: model and semantics, Appendix C: "media" keyword values*, 2000, http://www.ietf.org/rfc/rfc2911.txt
- *Grafica; scienza, tecnologia e arte della stampa e della comunicazione*, Arti poligrafiche europee
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4
- M. Fazio, *Dizionario e manuale delle unità di misura*, Zanichelli
- Marco Gaiarin, *Linux Italian HOWTO*, http://oss.sgi.com/LDP/HOWTO/Italian-HOWTO.html
- Maurizio Pistone, *Lingua italiana e altra linguistica*, http://www.mauriziopistone.it/testi/linguaitaliana.html
- Marco Baroni, Eros Zanchetta, *Morph-it! A free morphological lexicon for the Italian Language*, http://sslmitdev-online.sslmit.unibo.it/linguistics/morph-it.php
- *Amiga Translators' Organization*, http://bilbo.di.unipi.it/~ato-it/
- Bureau International des Poids et Mesures, http://www.bipm.org/
- Bureau International des Poids et Mesures, *Le Système international d'unités (SI)*, http://www1.bipm.org/utils/en/pdf/brochure-si.pdf

- Bureau International des Poids et Mesures, *The International System of Units (SI)* (traduzione in inglese), http://www1.bipm.org/utils/en/pdf/si-brochure.pdf
- National Institute of Standards and Technology, *International System of Units (SI)*, http://physics.nist.gov/Pubs/SP330/sp330sl30.pdf
- National Institute of Standards and Technology, *Guide for the Use of the International System of Units (SI)*, 1995, http://physics.nist.gov/cuu/pdf/sp811.pdf
- Markus Kuhn, *Standardized Units for Use in Information Technology*, 1995, http://www.cl.cam.ac.uk/~mgk25/information-units.txt
- National Institute of Standards and Technology, *Prefixes for binary multiples*, http://physics.nist.gov/cuu/Units/binary.html
- *OneLook dictionary search*, http://www.onelook.com/
- Denis Howe, *FOLDOC: free on-line dictionary of computing*, http://foldoc.org/
- Jukka Korpela, *A tutorial on character code issue*, http://www.cs.tut.fi/~jkorpela/chars.html
- *Unicode Home Page*, http://www.unicode.org/
- *Frequently asked questions*, http://www.unicode.org/faq/
- Ken Whistler, Mark Davis, *Unicode Technical Report #17, Character Encoding Model*, http://www.unicode.org/reports/tr17/

- C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin, P. Svanberg, *RFC 2130: The Report of the IAB Character Set Workshop held 29 February - 1 March, 1996*, http://www.ietf.org/rfc/rfc2130.txt
- Markus Kuhn, *UTF-8 and Unicode FAQ for UNIX/Linux*, http://www.cl.cam.ac.uk/~mgk25/unicode.html
- Bruno Haible, *The Unicode HOWTO*, http://tldp.org/HOWTO/Unicode-HOWTO.html
- D. Goldsmith, M. Davis, *RFC 2152: UTF-7, a mail-safe transformation format of Unicode*, 1997, http://www.ietf.org/rfc/rfc2152.txt
- F. Yergeau, *RFC 2279: UTF-8, a transformation format of ISO 10646*, 1998, http://www.ietf.org/rfc/rfc2279.txt
- Indrek Hein, *An online character database*, http://www.eki.ee/letter/
- Jukka Korpela, *The ISO Latin 1 character repertoire - a description with usage and notes*, http://www.cs.tut.fi/~jkorpela/latin1/
- Roman Czyborra, *The ISO 8859 Alphabet Soup*, http://czyborra.com/charsets/iso8859.html
- Roman Czyborra, *Codepages & Co.*, http://czyborra.com/charsets/codepages.html
- *Character Sets*, http://utopia.knoware.nl/users/eprebel/Communication/CharacterSets/

- IBM, *National language support guide and reference*, http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.nls/doc/nlsgdrf/nlsgdrf.pdf
- *Unicode character database*, http://www.unicode.org/Public/UNIDATA/
- *DICT.org*, http://www.dict.org
- *RFC 2229: A dictionary server protocol*, 1997, http://www.ietf.org/rfc/rfc2229.txt

[1] Questi tre stili sono molto importanti, in parte per motivi storici, ma soprattutto perché sono quelli che si hanno a disposizione più di frequente.

[2] Alcuni sistemi di composizione riescono a trarre il corsivo e il neretto da stili che per loro natura non hanno tali varianti. Per ottenerlo si utilizzano tecniche di deformazione e di trascinamento. In generale sarebbe bene evitare di sfruttare tali possibilità, dal momento che se uno stile non dispone di una serie, significa che non è adatto per quella.

[3] La tipizzazione di un carattere è l'azione con cui lo si rende uniforme.

[4] Questa sezione va considerata solo come un riferimento essenziale alla definizione di uno stile letterario e il contenitore di una piccola raccolta di regole, con le quali semplificare la vita di chi scrive documenti elettronici. Pur senza una competenza specifica sulla questione, viene introdotto l'argomento sottolineando alcuni concetti importanti.

[5] Secondo una regola della tipografia del passato, ormai condannata generalmente, è necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

[6] Purtroppo LaTeX segue la vecchia regola dell'allungamento dello spazio dopo il punto fermo che chiude il periodo, con l'aggravante che per riuscire a determinarlo può fare solo delle supposizioni, che a volte sono errate. Per fare in modo che LaTeX eviti di applicare questa regola errata, si può utilizzare il comando '`\frenchspacing`' nel preambolo del documento.

[7] Quando il sistema di composizione si basa su TeX e si usano virgolette elevate, le virgolette doppie si ottengono preferibilmente attraverso una coppia di apici singoli aperti (‘``’) e una coppia di apici singoli chiusi (‘''’). In altri casi, soprattutto quando si tratta di file di testo puri e semplici, gli apici doppi si indicano con le virgolette normali (‘"…"’).

[8] TeX permette l'uso di tre trattini di lunghezza differente: il trattino corto che si ottiene con un trattino singolo, il trattino medio che si ottiene con due trattini in sequenza e il trattino lungo che si ottiene con tre. Nella lingua italiana vanno usati solo i primi due, dove il trattino medio di TeX corrisponde al trattino lungo della grammatica italiana.

[9] Nell'ambito della documentazione tecnica, sarebbe consigliabile di evitare l'uso di accentazioni non comuni, anche se queste potrebbero essere preferibili in contesti più raffinati.

[10] Naturalmente questo ha senso se poi il programma di

composizione non tenta di suddividere le parole in sillabe.

[11] «Ciò che si vede è ciò che si ottiene»

[12] Trattando di oggetti grafici astratti, sembra più opportuno parlare di «simboli», piuttosto che di segni; in quanto è la composizione a trasformarli in segni.

[13] Nel linguaggio C, si distingue tra un tipo «carattere» tradizionale, corrispondente al byte, e un tipo ***carattere esteso***, ovvero *wide char*. Di conseguenza ci sono sia stringhe tradizionali, sia ***stringhe estese***, ovvero *wide string*.

[14] In generale, per maggiore chiarezza, i punti di codifica dell'Unicode e di ISO 10646 si indicano nella forma '**`U+`*nnnn***', oppure '**`U-`*nnnnnnnn***', dove ***n*** è una cifra esadecimale; ma come è già stato mostrato, qui si usa la notazione '**`#x`*n***' dell'XML.

[15] **Ascii** GNU GPL

[16] **Unicode** GNU GPL

[17] **Gucharmap** GNU GPL

[18] **Recode** GNU GPL

[19] **Iconv** GNU GPL

[20] **Ispell** software libero con licenza speciale

[21] Quando si ha a che fare con documentazione tecnica, in cui l'uso di termini in inglese è frequente, si potrebbe addirittura valutare la possibilità di basare l'analisi sul dizionario standard ('`english.hash`'), affiancando il dizionario personale specifico per il documento, solo che in tal caso si avrebbero difficoltà con le lettere accentate, dal momento che queste non sono previste nel file *affix* inglese.

[22] Le lettere '**ÿ**' e '**ß**', corrispondenti ai codici '**\377**' e '**\337**', sono minuscole e non hanno un equivalente maiuscolo nella codifica ISO 8859-1.

[23] In pratica, '**munchlist**' elimina queste parole ritenute estranee. Se si dispone di un elaboratore ben equipaggiato, si può dare in pasto a Ispell il file ottenuto dopo il riordino; lasciando a Ispell stesso il compito di eliminare i doppioni.

[24] **Aspell** GNU LGPL

[25] **Hunspell** GNU GPL, o GNU LGPL, o MPL

[26] **Textchk** GNU GPL

[27] **Dictd** GNU GPL

[28] **Serpento** GNU GPL

[29] **Dict** GNU GPL