Capitolo 14

Terminali a caratteri

14.1 Introduzione alla tastiera

La tastiera per elaboratore usata comunemente deriva storicamente da quella delle macchine da scrivere, attraverso l'esperienza della telescrivente. Il senso dell'organizzazione della tastiera e di alcune sue particolarità si perde se non si ripercorre brevemente la storia della scrittura meccanica.

Alcune delle foto che appaiono in questo capitolo sono state ottenute da altri documenti; in particolare, per le foto che non sono da ritenere di dominio pubblico è stato chiesto il permesso agli autori rispettivi.

I riferimenti alla fonte delle foto che non sono originali appaiono nelle didascalie delle stesse.

14.1.1 Funzionamento essenziale della macchina da scrivere tipica

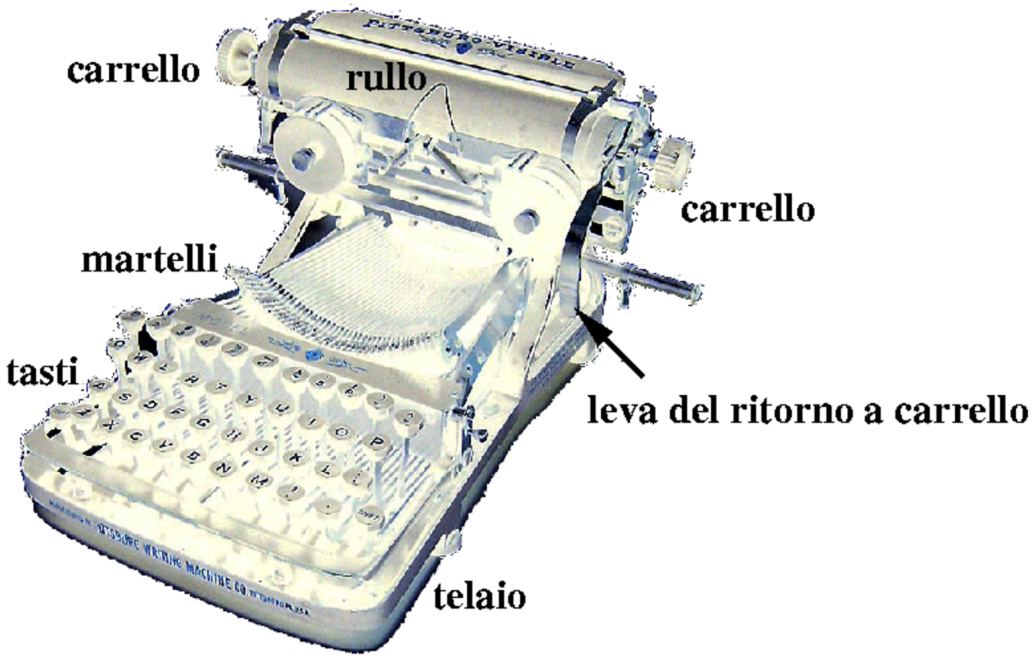
«

La macchina da scrivere meccanica tipica, usata fino agli anni 1950, è composta essenzialmente da un telaio, un carrello contenente un rullo sul quale si appoggia il foglio di carta, un blocco di martelli azionati da una tastiera. La pressione di un tasto comporta l'azionamento del martello corrispondente che va a colpire il foglio di carta che poggia sul rullo e quindi l'avanzamento del carrello in modo che la pressione di un nuovo tasto porti alla scrittura di un carattere nella posizione successiva, sulla stessa riga. Quando il carrello arriva a fine corsa, oppure quando viene raggiunto il margine destro di scrittura, si deve far avanzare il foglio di una riga e il carrello alla sua posizione iniziale, in modo da poter scrivere dopo il margine sinistro del foglio; per questo si usa normalmente una leva, azionando la quale si porta il testo a capo; l'azione in sé era nota come «ritorno a carrello».

Figura 14.1. Macchina da scrivere Pittsburg n. 10, probabilmente del 1902. Questa foto proviene da: Richard Polt, *The classic typewriter page*, http://site.xavier.edu/polt/typewriters/ .

Nella figura 14.2 si possono vedere evidenziati i componenti principali di una macchina da scrivere. In particolare, in questo caso la leva per il ritorno a carrello si trova sulla destra del carrello, mentre nella figura 14.3 si trova alla sua sinistra.

Figura 14.2. Componenti principali di una macchina da scrivere tradizionale.



14.1.2 Caratteri maiuscoli, minuscoli e altri simboli

La macchina da scrivere tradizionale dispone di un numero limitato di tasti, rispetto ai simboli tipografici che può scrivere; pertanto, si usano dei tasti da usare in combinazione. In generale si tratta di uno o due tasti equivalenti per ottenere le maiuscole, ma in casi particolari è presente anche un altro tasto per ottenere simboli non alfabetici (numeri e simboli di punteggiatura).

Nella macchina da scrivere a martelli, questi contengono due o tre ***tipi*** alfabetici, numerici, o di altri simboli: a seconda di come raggiungono il foglio di carta si ottiene la stampa di questo o di quel simbolo. In pratica, in alcune macchine da scrivere, il tasto usato per le maiuscole solleva il carrello in modo che i martelli lo colpiscano con la forma del simbolo posta nella parte superiore degli stessi; in altre macchine da scrivere, è il cesto dei martelli che si abbassa. Na-

turalmente, quando la macchina da scrivere ha meno tasti del solito, esiste la possibilità di regolare il sollevamento del carrello o l'abbassamento del cesto dei martelli a un livello intermedio, in modo che i martelli colpiscano con la forma di un simbolo centrale ulteriore.

Figura 14.3. Macchina da scrivere Fox portable n. 1, del 1917. Questa foto proviene da: Richard Polt, *The classic typewriter page*, http://site.xavier.edu/polt/typewriters/ .

La figura 14.3 mostra un esempio di macchina da scrivere che prevede tre livelli: minuscole, maiuscole e altri simboli. Si può leggere su un tasto a sinistra la scritta «CAP», a indicare che si tratta del tasto [*Maiuscole*], mentre più in alto, un altro tasto di cui non si legge la scritta, serve a bloccare lo spostamento (del carrello o del cesto dei martelli), in modo da avere sempre lettere maiuscole (da cui il nome «fissamaiuscole»,). Tra questi due tasti, ne appare uno con la scritta «FIG», che serve a ottenere i simboli tipografici non alfabetici

(l'opposto di *fig* sarebbe *ltr*).

Nelle primissime macchine da scrivere, lo spostamento dei martelli non era previsto e a ogni tasto corrispondeva un solo carattere. Nella figura 14.4 si vede un modello del genere, dove si può osservare che il gruppo di martelli forma proprio un cesto, a giustificarne la definizione.

Figura 14.4. Macchina da scrivere Calligraph 2, approssimativamente del 1888. Questa foto proviene da: Richard Polt, *The classic typewriter page*, http://site.xavier.edu/polt/typewriters/ .

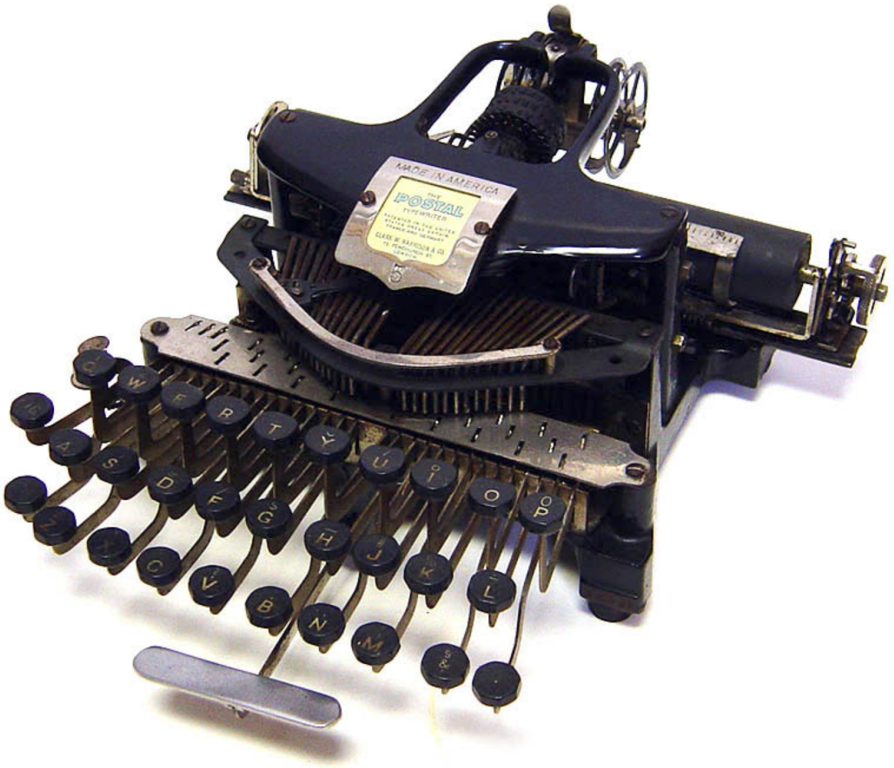
14.1.3 Disposizione anatomica dei tasti

«

La macchina da scrivere tradizionale tipica ha i tasti disposti in tre o quattro file, sfasate tra di loro. Ciò deriva sia da un'esigenza tecnica, per poter collocare in modo semplice le leve dei tasti, sia da un'esigenza anatomica. La figura 14.5 mostra una macchina da scrivere in

cui le leve dei tasti e l'esigenza meccanica a cui si è appena accennato risulta evidente. Si osservi che questa macchina, a differenze di tante altre, utilizza un cilindro di ***tipi*** al posto dei martelli.

Figura 14.5. Macchina da scrivere Postal 5, del 1904. Questa foto proviene da: Richard Polt, *The classic typewriter page*, *http://site.xavier.edu/polt/typewriters/* .



La figura 14.6 mostra una macchina da scrivere con una conformazione della tastiera abbastanza particolare, tale da ricordare quella di un pianoforte, disposta su un semicerchio.

Figura 14.6. Macchina da scrivere Hammond 1, approssimativamente del 1889. Questa foto proviene da: Richard Polt, *The classic typewriter page*, http://site.xavier.edu/polt/typewriters/ .

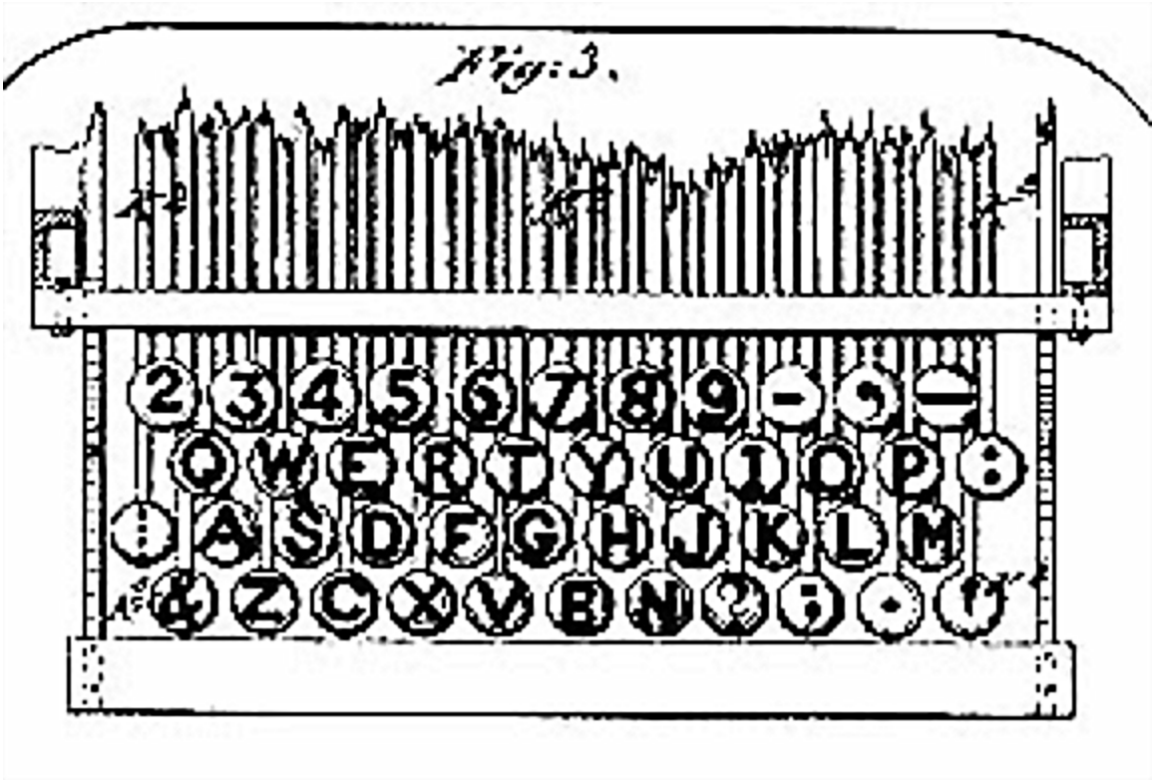
14.1.4 Mappa dei tasti

La prima produzione di macchine da scrivere negli Stati Uniti è quella progettata da Sholes e Glidden, prodotta da Remington, negli anni dal 1874 al 1878. Il progetto di questa macchina, dopo la definizione della sua struttura, ha compreso anche uno studio sulla disposizione dei simboli sui tasti, allo scopo di ridurre al minimo la possibilità di scontro tra i martelli usati per la scrittura. In pratica, due lettere in sequenza dovevano trovarsi a una distanza sufficiente da evitare l'incontro tra un martello che va e uno che viene, consentendo così di ridurre i tempi morti e di scrivere più velocemente, in base alle

caratteristiche di quella macchina.

Anche la disposizione dei tasti di quella macchina è stata brevettata, precisamente nel 1878, e il disegno lo si può vedere nella figura 14.7. Dal momento che i primi sei caratteri alfabetici compongono la sequenza «QWERTY», questo è diventato il nome di tale disposizione di tasti.

Figura 14.7. Disegno del brevetto della tastiera QWERTY del 1878. Questa foto, di dominio pubblico, è tratta da: Darryl Rehr, *The QWERTY connection*, http://home.earthlink.net/~dcrehr/ .



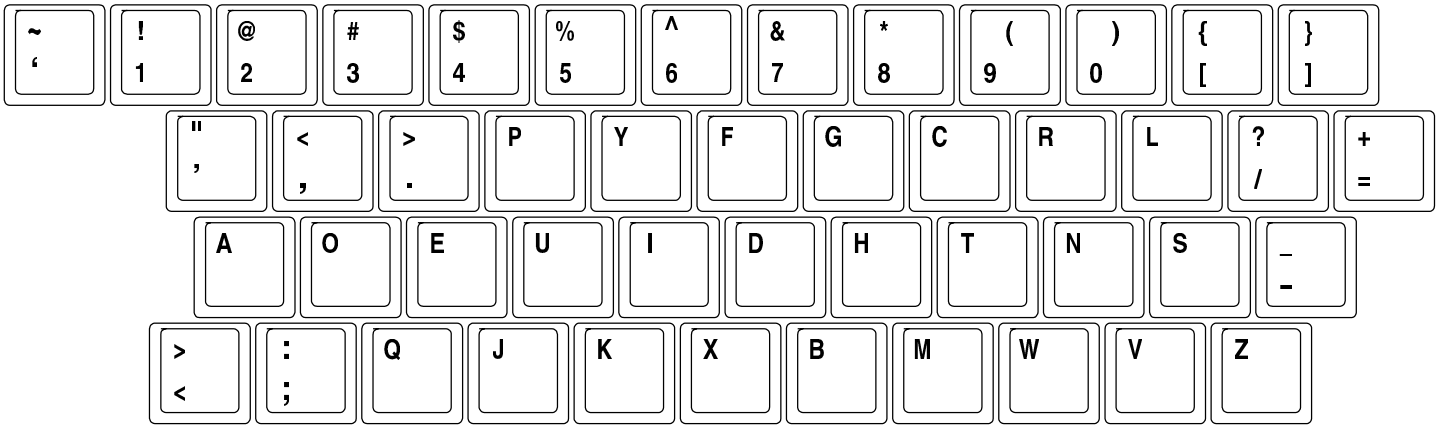
La maggior parte delle macchine da scrivere prodotte successivamente dai concorrenti ha utilizzato una mappa molto simile, almeno per quanto riguarda la porzione alfabetica.

La disposizione dei tasti così diversa dall'ordine alfabetico ha creato un mito attorno alle intenzioni reali del suo progettista; tuttavia il problema dello scontro tra i martelli in fase di scrittura esisteva ed

è esistito su tutte le macchine da scrivere con quel tipo di struttura. Pertanto, la motivazione per la quale una mappa del genere avrebbe lo scopo di accelerare la scrittura, è realistica, sulla base di quei limiti tecnici. Evidentemente, il motivo per il quale, quasi tutte le mappe di tastiere realizzate per le lingue che usano un alfabeto latino si rifanno fondamentalmente alla disposizione della QWERTY, rimane ormai dovuto alla consuetudine, dato che la scrittura elettronica non comporta più i problemi meccanici di quel tempo.

Nonostante la comodità della consuetudine, ci sono stati molti tentativi di proporre disposizioni differenti dei tasti, nell'ambito dell'alfabeto latino, fino al 1932, quando il professor August Dvorak propose una mappa studiata con criteri opposti rispetto alla QWERTY: i simboli usati con maggiore frequenza si trovano sulla riga di tasti su cui si posano inizialmente le dita. L'efficacia reale della mappa Dvorak rispetto alla QWERTY è controversa, così la mappa QWERTY e le sue varianti continuano a rappresentare lo standard.

Figura 14.8. Schema della mappa Dvorak.



La produzione di macchine da scrivere dei paesi europei ha seguito più o meno gli standard di fatto introdotti dagli Stati Uniti, anche per quanto riguarda la mappa dei tasti, salvo qualche piccola variante, soprattutto per l'esigenza di introdurre le lettere accentate. Le

differenze più importanti, a questo proposito, riguardano la collocazione delle cifre numeriche, le quali, a seconda dei casi, compaiono al livello delle lettere minuscole, oppure al livello delle lettere maiuscole.

Figura 14.9. Olivetti Lettera 35 del 1972, con mappa italiana QZERTY. Le cifre numeriche si ottengono quando si selezionano le maiuscole.

Figura 14.10. Tastiera di una macchina Hermes Media 3, con mappa tedesca QWERTZ. Le cifre numeriche si ottengono con il livello delle lettere minuscole.

14.1.5 Fissamaiuscole

«

La scrittura attraverso un numero limitato di tasti comporta nella macchina da scrivere la presenza di almeno un tasto per cambiare il ***livello***. Generalmente si tratta di soli due livelli, dove nel primo si collocano le lettere minuscole e nel secondo quelle maiuscole. Nelle macchine da scrivere si è arrivati ad avere due tasti [*Maiuscole*] equivalenti (⇧), da azionare con il mignolo della mano opposta a quella che deve scegliere la lettera o il simbolo da scrivere, assieme a un [*Fissamaiuscole*] (⇬), ovvero un tasto che mantiene inserita la selezione del livello delle maiuscole.

Date le limitazioni della tecnologia, il tasto [*Fissamaiuscole*] interveniva su tutti i tasti, anche su quelli che non servono per rappresentare segni alfabetici puri e semplici.

Tra i tasti che non servono per scrivere lettere alfabetiche ci sono, tra gli altri, quelli che servono per scrivere cifre numeriche; pertanto, la scelta della collocazione di questi simboli nel livello delle lettere minuscole o in quello delle lettere maiuscole, dipendeva dalla frequenza di utilizzo. La mappa italiana della tastiera ha ospitato le cifre numeriche nel livello delle lettere maiuscole, come si può vedere bene nella figura 14.9.

Con l'introduzione dei primi elaboratori dotati di terminali abbastanza completi (di sicuro già dalla fine degli anni 1970) e probabilmente anche delle prime macchine da scrivere elettroniche, negli Stati Uniti il tasto [*Fissamaiuscole*] ha cominciato a funzionare, precisamente, solo per le maiuscole, senza interferire con gli altri segni tipografici.

Nei paesi europei, generalmente, la diffusione degli elaboratori degli anni 1980 non ha portato alla modifica del funzionamento del [*Fissamaiuscole*] rispetto alle convenzioni costruite con le macchine da scrivere tradizionali. Questo lo si può verificare nel sistema operativo Dos, nel quale la configurazione della tastiera di paesi come Francia e Germania mantiene un funzionamento «tradizionale» del [*Fissamaiuscole*].

Si osservi che un sistema GNU/Linux tipico non consente la configurazione della tastiera in modo da far agire il tasto [*Fissamaiuscole*] su tutta la tastiera. Pertanto, alcune configurazioni locali della mappa della tastiera hanno richiesto un adattamento.

14.1.6 Avanzamento, arretramento e composizione

«

Nella maggior parte delle macchine da scrivere tradizionali lo spazio orizzontale si ottiene attraverso quella che è nota come «barra spaziatrice». Nella macchina da scrivere, il compito della barra spaziatrice è quello di fare avanzare il carrello, senza scrivere alcunché, al contrario di ciò che avviene di solito con la scrittura elettronica, dove invece si ottiene la «scrittura» di uno spazio (␣), mentre lo spostamento della posizione della scrittura avviene tramite un cursore controllato normalmente attraverso dei tasti muniti di frecce.

Con la comparsa delle macchine da scrivere a «scrittura visibile», dove il dattilografo poteva vedere immediatamente il testo che stava digitando, comincia ad apparire un tasto per l'arretramento. Anche in questo caso, lo scopo è solo quello di spostare il carrello all'indietro, rispetto al verso di scrittura. Questo arretramento poteva servire per correggere uno spostamento in avanti eccessivo, per tentare delle correzioni o per creare dei simboli composti (per esempio una virgola sovrapposta a una lettera «c», poteva servire per costruire una «ç», così come un apostrofo sovrapposto a una vocale poteva dare l'idea di un accento). Nella lingua inglese, questo tasto viene individuato con il termine *back space*.

La correzione di una digitazione errata richiedeva inizialmente l'uso di gomme per cancellare speciali: molto sottili, abrasive e tondeggianti. Dopo la cancellazione si poteva tornare sulla stessa posizione per riscrivere il testo corretto. In tempi più recenti si è introdotto l'uso di liquidi bianchi coprenti e di foglietti correttori. In questo ultimo caso, occorreva arretrare sulla posizione del carattere stampato erroneamente, inserire il foglietto di correzione e ribattere la stessa

lettera o lo stesso simbolo per cancellarlo. Un altro modo di «correggere», in mancanza di altro, ma soprattutto in mancanza di tempo, consisteva nel sovrascrivere il segno '**=**'; con lo stesso criterio, nel nastro perforato della telescrivente, il codice di controllo usato per indicare qualcosa di cancellato, o qualcosa senza valore, si otteneva rappresentando tutti i fori possibili. Infatti, seguendo la tradizione, nel codice ASCII, il carattere *<DEL>* si rappresenta come $7F_{16}$, pari a 1111111_2 (␡).

Con l'introduzione della scrittura elettronica, l'arretramento poteva assumere significati differenti, a seconda del metodo usato per la scrittura: poteva trattarsi di un arretramento puro e semplice, consentendo così la composizione di simboli sovrapposti, oppure di una cancellazione, quando il testo inserito veniva prima accumulato in memoria fino al termine della riga. Nel primo caso, la macchina da scrivere poteva disporre di un tasto speciale che cancellava il carattere premuto in precedenza in quella posizione.

L'evoluzione delle tecniche di correzione nella macchina da scrivere giustifica la presenza di diversi tasti nella tastiera per elaboratore. Il tasto noto come [*Backspace*] serve normalmente per arretrare cancellando (⌫); il tasto [*Canc*] (ovvero [*Del*] nelle tastiere inglesi) serve a cancellare il carattere che appare in corrispondenza del cursore, avvicinando il testo che dovesse trovarsi alla destra del cursore stesso; i tasti con le frecce spostano il cursore, ovvero la posizione in cui si intende andare a scrivere.

Come già accennato, nelle macchine da scrivere tradizionali, l'arretramento serviva anche per comporre assieme più simboli. Nella scrittura elettronica, soprattutto nel caso degli elaboratori, la composizione viene gestita in modo diverso, attraverso la definizione di

«tasti morti», i quali servono a dichiarare l'inizio di una sequenza di composizione che il software poi deve tradurre nel carattere corretto.

Nella tradizione italiana, con la scomparsa delle macchine da scrivere meccaniche si smette di pensare al concetto di composizione, tramite la sovrapposizione dei simboli. Infatti, la mappa per la tastiera apparsa nel 1983 nell'elaboratore Olivetti M24, non prevede la presenza di tasti morti. Tuttavia, il sistema operativo Dos di allora consentiva di generare altri simboli attraverso combinazioni con la tastiera numerica, pertanto non se ne avvertiva il problema. Al contrario, in altri paesi europei la configurazione della tastiera prevedeva spesso anche dei tasti morti per la composizione.

Per motivi tecnici, nella macchina da scrivere tradizionale il concetto di «inserimento» è assente del tutto, mentre compare solo nella scrittura elettronica, dove invece si può distinguere tra una scrittura che inserisce del testo, rispetto a una scrittura che sovrascrive eventualmente il testo che si trova in corrispondenza del cursore (a).

14.1.7 Tabulatore

«

A un certo punto dell'evoluzione delle macchine da scrivere tradizionali, si sente il bisogno di facilitare la scrittura di informazioni incolonnate. Per fare questo viene introdotta la tabulazione, che si ottiene fissando degli ***stop di tabulazione***, a cui si accede direttamente premendo un tasto: il ***tabulatore***.

In pratica, era necessario spostare orizzontalmente il carrello nella posizione desiderata, fissare lo stop di tabulazione e ripetere l'operazione per tutti gli altri incolonnamenti da programmare. Successi-

vamente, durante la scrittura, bastava premere il tabulatore per raggiungere immediatamente lo stop successivo. Per togliere gli stop programmati in precedenza, ci si doveva riposizionare negli stessi punti, usando un altro tasto per toglierli.

Nella scrittura elettronica la tabulazione di evolve, ma rimane il concetto di fondo (≒).

14.1.8 Telescrivente

«

La telescrivente è letteralmente la macchina che consente la scrittura a distanza, che nasce come evoluzione della telegrafia. La telescrivente costituisce anche la prima fase di adattamento della macchina da scrivere a terminale di un elaboratore elettronico. Il primo sistema telescrivente ragionevolmente efficace è quello di IBM, con il sistema Radiotype, del 1931.

Figura 14.11. Disegno della macchina da scrivere Blickensderfer Electric, del 1902. La prima macchina da scrivere elettrica, per di più con testina di scrittura cilindrica, apre la strada all'idea della telescrivente. Questa immagine proviene da: Richard Polt, *The classic typewriter page*, http://site.xavier.edu/polt/typewriters/ .

Rispetto alla macchina da scrivere, nella tastiera della telescrivente compaiono tasti con funzioni nuove, come il tasto [*Ctrl*] (⎈), con lo scopo di produrre codici di controllo che non sono associati alla scrittura di alcun simbolo. Attualmente, la storia della telescrivente sopravvive nel codice ASCII.

Generalmente, i codici di controllo necessari al funzionamento corretto delle comunicazioni attraverso una telescrivente sono generati quasi sempre attraverso combinazioni con un tasto, denominato *control*, con qualche eccezione eventuale per i codici più importanti, come per esempio <*ESC*> (⎋).

Tradizionalmente, con la telescrivente, l'associazione del tasto *control* si rappresenta con un accento circonflesso, secondo una forma del tipo '**^*x***'. Questa notazione rimane valida, purché utilizzata secondo il suo significato originale. In pratica, si deve considerare che le tastiere di un elaboratore comune si possono riconfigurare; pertanto, per fare un esempio, scrivere <^*a*> significa fare riferimento al codice ASCII 01_{16}, pari a <*SOH*>, ma non è detto, necessariamente, che per ottenere questo codice si debba premere sulla tastiera di oggi una combinazione del tipo [*Ctrl a*].

Tabella 14.12. Codici di controllo «C0», con la descrizione originale ottenuta da: *C0 control characters set*, 1987, *http://www.itscj.ipsj.or.jp/ISO-IR/140.pdf* .

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
0_{16}	NUL		NULL	A control character used to accomplish media-fill or time-fill. NUL characters may be inserted into or removed from, a stream of data without affecting the information content of that stream, but such action may affect the information layout and/or the control of equipment.
1_{16}	SOH	<^*a*>	START OF HEADING	A transmission control character used as the first character of a heading of an information message.
2_{16}	STX	<^*b*>	START OF TEXT	A transmission control character which precedes a text and which is used to terminate a heading.
3_{16}	ETX	<^*c*>	END OF TEXT	A transmission control character which terminates a text.
4_{16}	EOT	<^*d*>	END OF TRANSMISSION	A transmission control character used to indicate the conclusion of the transmission of one or more texts.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
5_{16}	ENQ	<^*e*>	ENQUIRY	A transmission control character used as a request from a remote station - the response may include station identification and/or station status. When a "Who are you" function is required on the general switched transmission network, the first use of ENQ after the connection is established shall have the meaning "Who are you" (station identification). Subsequent use of ENQ may, or may not, include the function "Who are you", as determined by agreement.
6_{16}	ACK	<^*f*>	ACKNOWLEDGE	A transmission control character transmitted by a receiver as an affirmative response to the sender.
7_{16}	BEL	<^*g*>	BELL	A control character that is used when there is a need to call for attention; it may control alarm or attention devices.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
8_{16}	BS	<^*h*>	BACKSPACE	A format effector which causes the active position to move one character position backwards.
9_{16}	HT	<^*i*>	HORIZONTAL TABULATION	A format effector which causes the active position to advance to the next pre-determined character position.
A_{16}	LF	<^*j*>	LINE FEED	A format effector which causes the active position to advance to the corresponding character position of the next line.
B_{16}	VT	<^*k*>	VERTICAL TABULATION	A format effector which causes the active position to advance to the corresponding character position on the next pre-determined line.
C_{16}	FF	<^*l*>	FORM FEED	A format effector which causes the active position to advance to the corresponding character position on a pre-determined line of the next form or page.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
D_{16}	CR	<^*m*>	CARRIAGE RETURN	A format effector which causes the active position to move to the first character position on the same line.
E_{16}	SO	<^*n*>	SHIFT OUT	A control character which is used in conjunction with SHIFT IN and ESCAPE to extend the graphic character set of the code. It may alter the meaning of octets 33 - 126 (dec.). The effect of this character when using code extension techniques is described in International Standard ISO 2022.
F_{16}	SI	<^*o*>	SHIFT IN	A control character which is used in conjunction with SHIFT OUT and ESCAPE to extend the graphic character set of the code. It may reinstate the standard meanings of the octets which follow it. The effect of this character when using code extension techniques is described in International Standard ISO 2022.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
10_{16}	DLE	<^*p*>	DATA LINK ESCAPE	A transmission control character which will change the meaning of a limited number of contiguously following bit combinations. Its is used exclusively to provide supplementary transmission control functions. Only graphic characters and transmission control characters can be used in DLE sequences.
11_{16}	DC1	<^*q*>	DEVICE CONTROL ONE	A device control character which is primarily intended for turning on or starting an ancillary device. If it is not required for this purpose, it may be used to restore a device to the basic mode of operation (see also DC2 and DC3), or for any other device control function not provided by other DCs.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
12_{16}	DC2	<^*r*>	DEVICE CONTROL TWO	A device control character which is primarily intended for turning on or starting an ancillary device. If it is not required for this purpose, it may be used to set a device to a special mode of operation (in which case DC1 is used to restore the device to the basic mode), or for any other device control function not provided by other DCs.
13_{16}	DC3	<^*s*>	DEVICE CONTROL THREE	A device control character which is primarily intended for turning off or stopping an ancillary device. This function may be a secondary level stop, for example wait, pause, stand-by or halt (in which case DC1 is used to restore normal operation). If it is not required for this purpose, it may be used for any other ancillary device control function not provided by other DCs.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
14_{16}	DC4	<^*t*>	DEVICE CONTROL FOUR	A device control character which is primarily intended for turning off, stopping or interrupting an ancillary device. If it is not required for this purpose, it may be used for any other device control function not provided by other DCs.
15_{16}	NAK	<^*u*>	NEGATIVE ACKNOWLEDGE	A transmission control character transmitted by a receiver as a negative response to the sender.
16_{16}	SYN	<^*v*>	SYNCHRONOUS IDLE	A transmission control character used by a synchronous transmission system in the absence of any other character (idle condition) to provide a signal from which synchronism may be achieved or retained between data terminal equipment.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
17_{16}	ETB	<^*w*>	END OF TRANSMISSION BLOCK	A transmission control character used to indicate the end of a transmission block of data where data is divided into such blocks for transmission purposes.
18_{16}	CAN	<^*x*>	CANCEL	A character, or the first character of a sequence, indicating that the data preceding it is in error. As a result, this data shall be ignored. The specific meaning of this character shall be defined for each application and/or between sender and recipient.
19_{16}	EM	<^*y*>	END OF MEDIUM	A control character that may be used to identify the physical end of a medium, or the end of the used portion of a medium, or the end of the wanted portion of data recorded on a medium. The position of this character does not necessarily correspond to the physical end of the medium.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
$1A_{16}$	SUB	<^z>	SUBSTITUTE	A control character used in the place of a character that has been found to be invalid or in error. SUB is intended to be introduced by automatic means.
$1B_{16}$	ESC	<^[>	ESCAPE	A control character which is used to provide additional characters. It alters the meaning of a limited number of contiguously following bit combinations. The use of this character is specified in ISO 2022.
$1C_{16}$	FS IS4	<^\>	FILE SEPARATOR INFORMATION SEPARATOR FOUR	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order it delimits a data item called a file.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
$1D_{16}$	GS IS3	<^]>	GROUP SEPARATOR INFORMATION SEPARATOR THREE	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order, it delimits a data item called a group.
$1E_{16}$	RS IS2	<^^>	RECORD SEPARATOR INFORMATION SEPARATOR TWO	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order it delimits a data item called a record.
$1F_{16}$	US IS1	<^_>	UNIT SEPARATOR INFORMATION SEPARATOR ONE	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order, it delimits a data item called a unit.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
$7F_{16}$	DEL		DELETE	A character used primarily to erase or obliterate an erroneous or unwanted character in punched tape. DEL characters may also serve to accomplish media-fill or time-fill. They may be inserted into, or removed from, a stream of data without affecting the information content of that stream, but such action may affect the information layout and/or the control of equipment.

Figura 14.13. La telescrivente più conosciuta dagli anni 1960. Questa foto proviene dalla raccolta di Peter Roosen-Runge, visibile alla pagina *York computing: scenes from a distant past*, http://www.cse.yorku.ca/~peter/deptphotos/photos.html

Figura 14.14. Mappa della tastiera della telescrivente Teletype ASR-33.

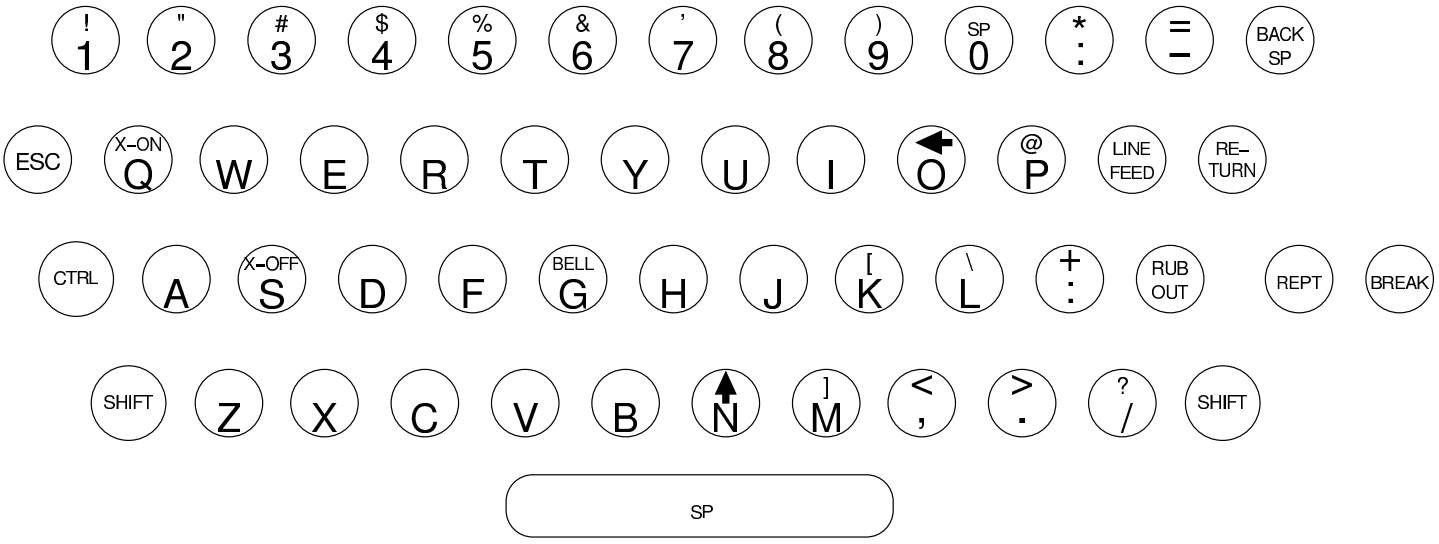


Figura 14.15. Il terminale video più comune, DEC VT100 del 1978, aderente allo standard ANSI X3.64, ovvero ECMA-48, ovvero ISO/IEC 6429. Questa foto è di Dave Dunfield, tratta da *Daves Old Computers* http://www.classiccmp.org/dunfield/tty/ .

14.1.9 Mutazione della mappa italiana

Gli anni 1980 rappresentano il momento in cui la macchina da scrivere comincia a cedere il posto agli elaboratori personali, un po' in tutto il mondo. Questo passaggio, per quanto riguarda l'Italia, segna la trasformazione della mappa italiana per le tastiere.

Alla fine degli anni 1970 cominciavano a diffondersi elaboratori professionali abbastanza completi. In quel periodo la produzione italiana era praticamente assente, o comunque non comparabile sul piano tecnico. Gli elaboratori in questione erano quindi importati dagli Stati Uniti o dalla Gran Bretagna, dotati di tastiere QWERTY senza lettere accentate. Tuttavia, lo standard di scrittura della lingua italiana prevede gli accenti quasi esclusivamente alla fine delle parole, cosa che si poteva ottenere graficamente con l'apostrofo: era il contesto a far capire se si doveva intendere come apostrofo, accento grave o accento acuto.

In un certo senso, in Italia si diffondeva l'idea che la tastiera per elaboratore fosse una cosa e che la tastiera della macchina da scrivere fosse un'altra.

Tra il 1982 e il 1983 succede qualcosa che trasforma la «tastiera italiana» in una QWERTY con lettere accentate, ma è difficile ricostruire i passaggi di questa mutazione. Infatti, in quel periodo Olivetti produceva elaboratori, proponendo nel mercato italiano, sia tastiere tradizionali QZERTY, sia tastiere QWERTY. Questa incertezza ha fine con la distribuzione del modello M24, dove appare una tastiera QWERTY adattata con le lettere accentate per la lingua italiana (figura 14.20). Probabilmente è proprio l'elaboratore Olivetti M24

il «responsabile», nel bene o nel male, della trasformazione della tastiera italiana.

Figura 14.16. Tastiera QWERTY dell'elaboratore Olivetti M20, anno 1982, da un opuscolo per il mercato italiano. Si osservi comunque che il modello M20 BC aveva una tastiera QZERTY tradizionale.

Figura 14.17. Tastiera QZERTY dell'elaboratore Olivetti M20 BC, anno 1982, da una foto di Davide Bucci.

Figura 14.18. Tastiera QZERTY dell'elaboratore Olivetti M40 BC, anno 1982, da un opuscolo.

Figura 14.19. Tastiera QZERTY dell'elaboratore portatile Olivetti M10, anno 1983. L'immagine è stata ottenuta da un'asta telematica di apparecchiature usate.

Figura 14.20. Tastiera QWERTY italiana dell'elaboratore Olivetti M24, anno 1983. Si osservi che il tasto [*Fissamaiuscole*] funziona solo nella porzione alfabetica della tastiera, escludendo anche le lettere accentate.

Nonostante la produzione da parte di Olivetti dell'elaboratore M24 con una nuova tastiera italiana, continua la sua produzione di macchine da scrivere per qualche tempo, ma sempre con la mappa tradizionale QZERTY, sottolineando la distinzione tra macchina da scrivere ed elaboratore.

Figura 14.21. Macchina da scrivere elettronica Olivetti ET116, probabilmente dell'anno 1987.

Figura 14.22. Dettaglio della tastiera della macchina da scrivere elettronica Olivetti ET116.

14.2 Tastiera di un elaboratore

«

La gestione della tastiera con un sistema operativo è generalmente una questione abbastanza complicata. Un sistema operativo destinato ad architetture differenti deve essere in grado di gestire tastiere fisiche molto diverse tra di loro, mantenendo un comportamento uniforme.

Per poter adattare tastiere fisiche differenti allo stesso sistema operativo, è necessario creare un'astrazione rispetto alla tastiera reale, abbinando la pressione dei tasti fisici a simboli di una tastiera virtuale che rimane sempre la stessa. La tastiera virtuale dei sistemi che si rifanno al modello di Unix, genera dei simboli, che possono avere il significato di un carattere, di una stringa, o di una funzione speciale. Quello che si fa sulla tastiera fisica è un'azione meccanica, ma quello che conta è ciò che, da quella azione, viene generato attraverso la tastiera virtuale. Per esempio, se ipoteticamente con la combinazione di tasti [*AltGr e*] si ottiene il simbolo «€», per il sistema è come se si premesse il tasto [€] della tastiera virtuale.

Nelle tastiere fisiche si utilizzano diversi tasti «modificatori» per generare delle combinazioni, per esempio per ottenere le lettere maiuscole, mentre, nella tastiera virtuale si può immaginare che ci sia un tasto singolo per ogni funzione.

Nella tastiera virtuale di un sistema simile a Unix si prevedono tasti con funzioni speciali, che in pratica trasmettono un carattere o una stringa che ha lo scopo di comunicare qualcosa a un programma. Per esempio, il codice ASCII <*BS*>, ovvero 08_{16}, ha normalmente lo scopo di cancellare un testo, arretrando; il codice ASCII <*STX*>, ovvero 02_{16}, potrebbe avere lo scopo di informare il programma della

conclusione del flusso di dati in ingresso (*end of file*). Per ottenere questi simboli con funzioni speciali si può immaginare la tastiera virtuale con un tasto apposito per ognuno, mentre nella tastiera reale potrebbe essere necessaria una combinazione (nel caso del codice di conclusione di un flusso di dati si usa normalmente la combinazione [*Ctrl d*]). Tuttavia, la tastiera virtuale tipica di un sistema Unix prevede dei modificatori (virtuali): si tratta precisamente di <*Control*> e <*Meta*>.

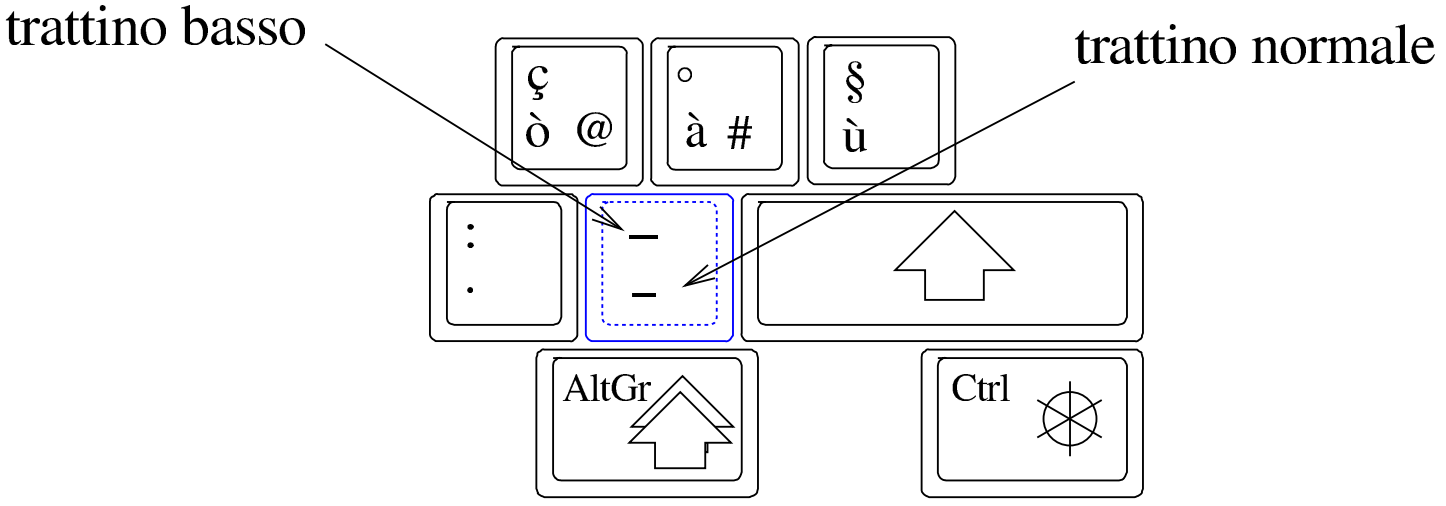
I modificatori virtuali <*Control*> e <*Meta*> servono per ottenere delle funzioni speciali, in abbinamento con altri tasti (virtuali). Tuttavia, si potrebbero immaginare tasti virtuali separati per ognuna delle combinazioni prestabilite, rendendo il concetto più semplice.

Il fatto che la tastiera virtuale di un sistema basato sul modello di Unix preveda dei modificatori virtuali, è fonte di confusione, perché, per esempio, da un lato può essere «facile» abbinare mentalmente il tasto reale [*Ctrl*] al modificatore virtuale <*Control*>, dall'altro, però, non è detto che la combinazione reale [*Ctrl* ***x***] produca effettivamente la selezione della funzione abbinata alla combinazione virtuale <*Control_**x***>. Per evitare questa confusione, sarebbe meglio pensare a funzioni associate a tasti virtuali indipendenti: <*Control_a*>, <*Control_b*>, <*Control_c*>,... <*Meta_a*>, <*Meta_b*>, <*Meta_c*>,...

Di norma, su una tastiera reale per elaboratori x86, si associa il modificatore virtuale <*Control*> ai tasti [*Ctrl*] presenti, in modo indifferente, mentre si associa normalmente il modificatore virtuale <*Meta*> al tasto [*Alt*] a sinistra della barra spaziatrice. Tuttavia, certe com-

binazioni virtuali non si possono ottenere sempre nello stesso modo con la tastiera reale; per esempio, con una tastiera italiana, la combinazione virtuale <*Control_underscore*> potrebbe richiedere la pressione simultanea di [*Ctrl Maiuscole* −] (control, maiuscole, trattino normale), perché il carattere '_' si ottiene in combinazione con il tasto delle maiuscole ([*Maiuscole* −]). D'altra parte, dal momento che non esiste una combinazione virtuale che abbina <*Control*>, con il segno '−', sempre ipotizzando una tastiera italiana, la combinazione virtuale <*Control_underscore*> potrebbe essere comandata da una combinazione reale differente, come potrebbe esserlo [*Ctrl* −], dato che il segno '−' si ottiene sullo stesso tasto reale, ma senza la pressione del tasto delle maiuscole. Tutto questo dipende comunque dalla configurazione della mappa della tastiera, con la quale si associa la pressione dei tasti reali ad azioni appropriate sulla tastiera virtuale del sistema operativo.

Figura 14.23. Porzione di tastiera italiana in cui si vede in particolare il tasto con il quale si ottengono i simboli '−' e '_'.



Riquadro 14.24. Notazioni e simbologia.

Nella documentazione tecnica a proposito di programmi per sistemi Unix, si fa spesso riferimento a combinazioni di tasti da premere, secondo una simbologia particolare. Di solito si usano notazioni del tipo '**^*x***', oppure '**C-*x***', per rappresentare una cosa che potrebbe essere descritta come «Ctrl+***x***»; così '**M-*x***', per rappresentare combinazioni del tipo «Meta+***x***». Quello che conta, a parte la simbologia usata, è comprendere che si fa riferimento a «combinazioni virtuali», che nella realtà potrebbero richiedere metodi di inserimento differenti rispetto all'apparenza della simbologia usata.

14.2.1 Livelli

«

Nelle tastiere fisiche comuni, i tasti a disposizione sono inferiori a quelli che servirebbero, per poter esprimere tutti i segni tipografici richiesti da una certa lingua. Nella macchina da scrivere meccanica si otteneva la duplicazione delle funzioni dei tasti, alzando o abbassando il cesto dei martelli. Per questa ragione, anche oggi si parla di ***livelli***, a proposito del fatto che un tasto consenta di ottenere due o più simboli differenti, combinandolo con un modificatore appropriato.

Nelle tastiere europee comuni si utilizzano almeno tre livelli, dove il secondo livello viene attivato con la pressione del tasto [*Maiuscole*], mentre il terzo livello si ottiene con un tasto alternativo (nel caso delle tastiere per elaboratori x86 si tratta normalmente del tasto [*AltGr*]). Per la disposizione italiana, il terzo livello serve per esempio a ottenere la chiocciola, il cancelletto e le parentesi quadre.

La gestione dei livelli è compito della configurazione della tastiera fisica e si possono ipotizzare anche più di tre livelli, con l'uso di

combinazioni più complesse.

Generalmente, il tasto [*Fissamaiuscole*] di una tastiera fisica non interviene su tutta la tastiera, ma solo su una porzione centrale. Questo fatto ha dei vantaggi in fase di digitazione, perché i tasti che non hanno una forma minuscola contrapposta a una maiuscola, non vengono coinvolti. Tuttavia, nelle convenzioni di alcuni paesi europei, il tasto [*Fissamaiuscole*] dovrebbe agire su tutta la porzione alfanumerica, pertanto, in quei casi, capita di avere le cifre numeriche al secondo livello.

Questo comportamento che consiste nell'avere un fissamaiuscole che interviene su tutta la tastiera alfanumerica, si può osservare nella gestione della tastiera di un sistema operativo Dos, in particolare con le mappe per le lingue francese e tedesca. Tali mappe hanno mantenuto una stretta corrispondenza con la tastiera della macchina da scrivere, tanto che lo sblocco del fissamaiuscole si ottiene premendo il tasto [*Maiuscole*].

È bene osservare che in una console GNU/Linux non è possibile configurare il funzionamento del tasto [*Fissamaiuscole*], pertanto la mappa delle tastiere per la lingua francese e per la lingua tedesca non corrispondono esattamente alla tradizione. Tuttavia, è da osservare che anche con X (il sistema grafico) non viene più rispettata la tradizione originale.

14.2.2 Alfabeti non latini

Quando si utilizza una tastiera per un terminale a caratteri, come nel caso della console, è normale che ci si trovi nella necessità di scrivere qualcosa utilizzando l'alfabeto latino, se non altro per impartire dei comandi al sistema operativo. Per questa ragione, se la lingua che

si intende usare si basa su un alfabeto non latino (come per il greco e il russo), non si può fare a meno di disporre di una tastiera che consenta di generare ugualmente simboli nell'alfabeto latino.

Nel caso della console di un sistema GNU/Linux, di solito si fa in modo che il tasto che porta al terzo livello ([*AltGr*], o qualunque altra cosa in base alla configurazione della mappa), funzioni come il [*Fissamaiuscole*], ovvero che mantenga lo spostamento al terzo livello fino a quando non lo si ripreme una seconda volta. In questi casi, di solito, i primi due livelli offrono una disposizione dei tasti uguale o simile a quella statunitense, mentre i due livelli successivi servono per l'alfabeto non latino da usare.

Si osservi che l'utilizzo di un alfabeto non latino implica l'uso di una codifica speciale. L'uso della codifica UTF-8 dovrebbe garantire la compatibilità con quella ASCII, ma se si usa ancora una codifica differente, la configurazione della mappa della tastiera della console di un sistema GNU/Linux deve tenerne conto.

14.2.3 Tasti morti e composizione

Due modi ulteriori per estendere le potenzialità di una tastiera fisica consistono nell'attribuire a una sequenza di tasti il compito di ***comporre*** un carattere unico. Di solito si fa questo per generare delle lettere accentate, digitando prima l'accento e poi la lettera da accentare.

Uno dei due modi consiste nell'avere dei simboli che servono ad accentare la lettera successiva; alla selezione di uno di questi accenti non si ottiene nulla fino alla selezione di un simbolo ulteriore da modificare. In questo caso, il simbolo selezionato inizialmente è

un ***tasto morto***, o un accento morto, in quanto da solo non produce nulla.

Un altro modo consiste nell'avere un tasto modificatore, premendo il quale si passa in modalità di composizione. In tale modalità si deve inserire una stringa che in qualche modo deve generare un simbolo appropriato. Di norma, al termine di una sequenza di composizione, la modalità di funzionamento torna a essere quella normale.

Queste modalità di funzionamento della tastiera (tasti morti e composizione) vengono definite attraverso la sua configurazione, ma ovviamente si possono ottenere solo le lettere, accentate o composte, previste espressamente.

14.2.4 Inserimento numerico del codice di un simbolo

«

Un modo ancora più completo per consentire l'inserimento di simboli che sulla propria tastiera non possono essere previsti, sta nel definire una combinazione che richiede l'inserimento di un numero, con il quale si identifica il punto di codifica del simbolo desiderato (per una definizione del punto di codifica si veda la sezione 47.6). Questo metodo di inserimento di simboli insoliti si è diffuso in particolare attraverso il sistema operativo Dos, con il quale era possibile premere il tasto [*Alt*] sinistro e digitare il numero del carattere desiderato con la tastiera numerica.

La convenzione introdotta dal sistema operativo Dos viene mantenuta generalmente anche nella configurazione della tastiera di un sistema GNU/Linux comune, ma si tratta di una funzione che non è sempre presente; per esempio manca quando si usa la grafica con X. A ogni modo, è bene considerare che ci possono essere meto-

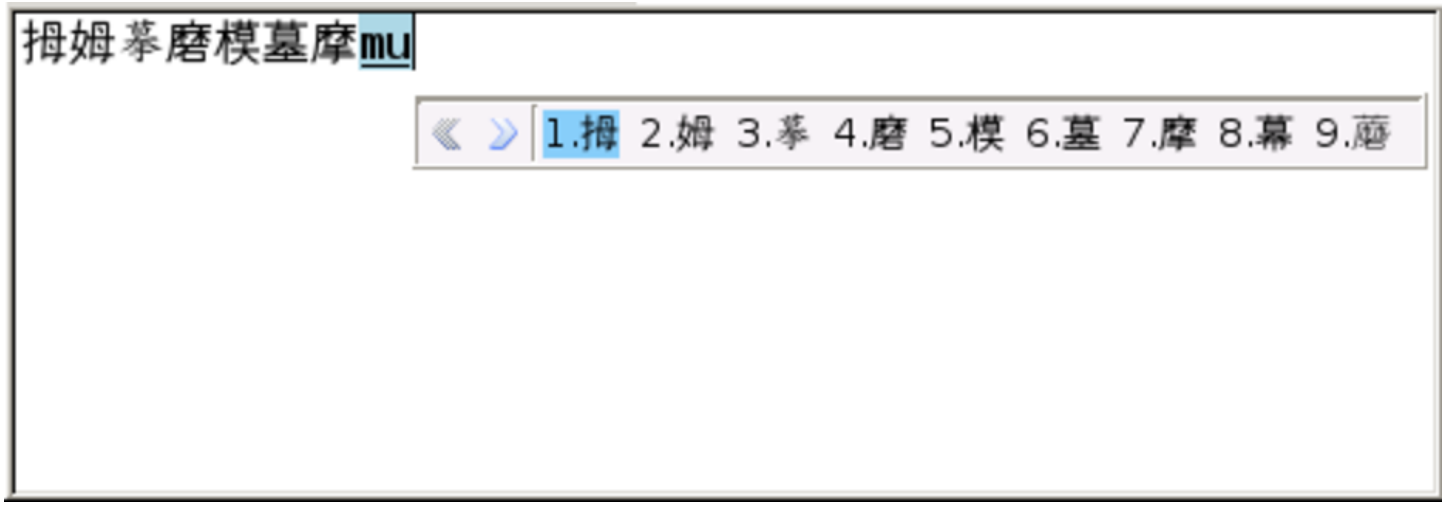
di alternativi di inserimento; per esempio potrebbe essere possibile specificare il numero del punto di codifica anche in esadecimale.

14.2.5 Alfabeti asiatici

Quando una lingua richiede la scrittura attraverso un numero elevato di caratteri (come il cinese), le tastiere comuni, che sono nate per l'alfabeto latino, diventano troppo limitate. In queste condizioni, l'utilizzo dei livelli, dei tasti morti e della composizione, diventa insufficiente o inadatto per poter gestire il volume di simboli che la lingua richiede. Pertanto, in queste situazioni, si utilizza generalmente un programma aggiuntivo, collocato idealmente dopo la tastiera astratta. In pratica, la configurazione della tastiera fisica potrebbe anche essere predisposta secondo la lingua inglese, mentre poi, un programma che si inserisce prima dell'applicazione utilizzata effettivamente consente di selezionare i simboli in base a tecniche alternative.

Il metodo di trasformazione dall'alfabeto latino ai simboli della lingua prescelta, avviene generalmente attraverso una qualche forma di traslitterazione, dove si arriva anche a dare un nome (utilizzando l'alfabeto latino) al carattere da scrivere. A questo proposito si usa la sigla «IM», *Input method*, che fa spesso parte dei programmi che si occupano di tale compito (per esempio SCIM e Xim).

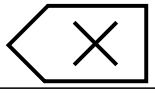
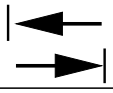
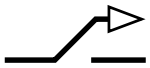
Figura 14.25. Un esempio si inserimento di un testo in cinese, attraverso un metodo di inserimento intelligente basato sul sistema di traslitterazione pinyin. In questo caso, si vede l'inserimento della parola «mu» (secondo la traslitterazione in caratteri latini), per la quale esistono tante varianti nel modo di rappresentarla. Per scegliere il carattere corretto, dopo l'inserimento del nome, si seleziona il numero corrispondente.



14.2.6 Pittogrammi ISO 9995-7

Lo standard ISO 9995-7 definisce alcuni simboli, ovvero ***pittogrammi***, da usare sulla tastiera per funzionalità comuni, in modo da evitare le descrizioni verbali che possono richiedere una traduzione. La tabella successiva riepiloga alcuni dei simboli in questione. Si osservi che parte dei pittogrammi previsti dallo standard ISO 9995-7 è inserito nell'insieme di caratteri universale; si tratta precisamente dei codici da U+2380 a U+238C, da U+2396 a U+239A e U+2425.

Tabella 14.26. Alcuni pittogrammi ISO 9995-7.

Simbolo	Descrizione	Simbolo	Descrizione
	[*Ins*]		[*Canc*]
	[*Backspace*]		[*Tab*]
	[*Maiuscole*]		passaggio al terzo livello
	[*Fissamaiuscole*]		[*BlocNum*]
	[*Ctrl*]		[*Alt*]
	[*Comp*] inizia una sequenza di composizione		[*Invio*]
	[*Interr*] *break*		[*Pausa*]
	[*Esc*]		[*Inizio*]
	[*Fine*]		[*pagina−su*] pagina precedente
	[*pagina−giù*] pagina successiva		[*Stampa*]
	[*BlocScorr*]		[*Spazio*]
	spazio non separabile		separatore decimale

14.3 Tastiera della console di un sistema GNU/Linux

La gestione della tastiera della console di un sistema GNU/Linux avviene in modo piuttosto complesso. Solitamente si fa riferimento al programma '**`loadkeys`**' come unico responsabile della defi-

nizione delle funzioni associate ai tasti, ma questo non basta per comprendere il problema.

I segnali della tastiera vengono ricevuti direttamente dal kernel che poi li fornisce ai programmi, in vario modo, a seconda di una data modalità selezionata. Il programma '**kbd_mode**' permette di conoscere o di modificare la modalità di funzionamento della tastiera, ma la modifica è da riservare solo a occasioni particolari, di solito utilizzando una connessione remota, quando un programma ha modificato la modalità della tastiera rendendo inutilizzabile la console.

14.3.1 Tastiera locale e tastiera remota

«

In generale, ciò che conta è fare in modo che funzioni correttamente **la tastiera connessa al proprio elaboratore**. Questo è importante perché, quando si utilizza una connessione remota, per esempio attraverso il protocollo TELNET, la tastiera che si adopera dipende, per la sua configurazione, dall'elaboratore a cui è connessa fisicamente: non dipende quindi dall'elaboratore al quale si è collegati. Leggendolo così potrebbe sembrare una cosa evidente, ma quando ci si trova a farlo veramente, non lo è più così tanto.

In questo senso, se da una connessione remota viene dato un comando per modificare la modalità di funzionamento o la mappa della tastiera, l'effetto si risente sulla console dell'elaboratore che riceve il comando e non nel terminale remoto.

Queste considerazioni permettono anche di comprendere che la connessione remota è indipendente da qualunque configurazione che riguardi la tastiera di un certo elaboratore. Perciò, una configurazione errata che renda inutilizzabile una console, può essere corretta attraverso una connessione remota.

14.3.2 Console virtuali

Un sistema GNU/Linux, come altri sistemi Unix, consente di gestire diversi terminali sull'unica console esistente effettivamente. Questi vengono definiti ***console virtuali***. Attraverso alcune combinazioni di tasti si riesce a passare da una console virtuale all'altra. Queste combinazioni sono normalmente [*Alt F1*], [*Alt F2*],... oppure [*Ctrl Alt F1*], [*Ctrl Alt F2*],... ma possono essere modificate (anche se ciò non è consigliabile).

La console vera e propria corrisponde quasi sempre alla console virtuale in funzione in un dato momento.

La configurazione della tastiera, a seconda del tipo di modalità su cui si interviene, può avere effetto su tutte le console virtuali, oppure solo su quella attiva.

14.3.3 Controllo della modalità di funzionamento della tastiera con «kbd_mode»

Il programma '**kbd_mode**'[1] permette di conoscere o di modificare la modalità di funzionamento della tastiera della console. Ciò significa, implicitamente, che l'effetto riguarda la console virtuale attiva; pertanto, quando viene utilizzato a distanza, attraverso il protocollo TELNET o un altro metodo di connessione simile, ha effetto sulla console virtuale attiva nell'elaboratore al quale si è connessi.

```
kbd_mode [opzioni]
```

L'utilizzo di questo programma deve essere fatto con prudenza: la visualizzazione della modalità di funzionamento della tastiera non

provoca alcun inconveniente, ma la modifica errata della modalità, comporta l'impossibilità di continuare a utilizzarla.

È meglio evitare di utilizzare questo programma per modificare la modalità della tastiera, da una finestra di terminale, all'interno del sistema grafico X.

Tabella 14.27. Alcune opzioni.

Opzione	Descrizione
	Se '**kbd_mode**' viene avviato senza opzioni, il risultato che si ottiene è la visualizzazione della modalità attiva. Ciò dovrebbe corrispondere all'uso normale del programma.
`-s`	L'opzione '**-s**' attiva la modalità *scancode*, o RAW. Questa è la modalità che si osserva normalmente nelle finestre di terminale del sistema grafico X. Questa modalità non può essere utilizzata per le console virtuali normali.
`-k`	L'opzione '**-k**' attiva la modalità *keycode*, o MEDIUMRAW. Questa modalità non può essere utilizzata per le console virtuali normali.
`-a`	L'opzione '**-a**' attiva la modalità ASCII o XLATE. Questa modalità è quella normale quando si utilizzano le console virtuali. Questa modalità fa uso della mappa definita da '**loadkeys**'.
`-u`	L'opzione '**-u**' attiva la modalità UTF-8 o UNICODE. Questa modalità fa uso della mappa definita da '**loadkeys**'.

Se la console di un elaboratore è rimasta bloccata e comunque esi-

ste la possibilità di connettersi a questo da un'altra postazione funzionante, si può ripristinare la modalità corretta della tastiera da lì. Nell'esempio seguente, l'elaboratore da sistemare è *dinkel.brot.dg* e quello dal quale si interviene è *roggen.brot.dg*.

```
roggen.brot.dg$ telnet dinkel.brot.dg [Invio]

  Trying dinkel.brot.dg...
  Connected to dinkel.brot.dg.
  Escape character is '^]'.

login: root [Invio]

Password: ******** [Invio]

dinkel.brot.dg# kbd_mode -a [Invio]

dinkel.brot.dg# exit [Invio]
```

Questo esempio presume che si possieda già una certa conoscenza di come si instaura una connessione remota attraverso un cliente TELNET. L'utente inesperto che dovesse tentare una cosa del genere potrebbe non essere capace di completare l'accesso a causa del fatto che normalmente viene impedito all'utente '**root**' di accedere da una postazione remota, per motivi di sicurezza.

14.3.4 Controllo sullo stato della tastiera con «setleds»

Il programma '**setleds**'[2] interviene esclusivamente su una console virtuale attiva, o meglio, quella da cui proviene lo standard input. Non può essere utilizzato in altre situazioni. Lo scopo del program-

ma è di intervenire sull'impostazione dei tasti [*Fissamaiuscole*] (*capslock*), [*BlocNum*] (*numlock*) e [*BlocScorr*] (*ScrollLock*).

```
setleds [opzioni] [modalità…]
```

Solitamente, questi tasti attivano o disattivano la modalità corrispondente, che viene segnalata anche da una spia luminosa sulla tastiera, una per ogni modalità. Queste spie sono notoriamente dei ***led*** (*Light emitting diode*) e spesso sono chiamati così anche in italiano.

Il programma permette di intervenire sia attivando o disattivando queste modalità, sia accendendo o spegnendo i led.

Tabella 14.29. Alcune opzioni e modalità

Opzione	Descrizione
	Utilizzando il programma senza argomenti, si ottiene il resoconto sull'impostazione dei tasti su cui può intervenire e su quella dei led corrispondenti.
`-F`	L'opzione '**`-F`**' è quella predefinita, quando vengono indicate solo delle modalità. Con questa, si modifica lo stato corrispondente alle modalità indicate. Se i led non sono impostati indipendentemente, riflettono la nuova situazione.
`-D`	L'opzione '**`-D`**' è analoga a '**`-F`**', con la differenza che la nuova impostazione diviene predefinita ed è ciò che si ripresenta a seguito di un ripristino attraverso l'uso del comando '**`reset`**'.

Opzione	Descrizione
`-L`	L'opzione '**`-L`**' fa in modo di intervenire solo sui led. Dal momento in cui si utilizza '**`setleds`**' con questa opzione, si sgancia il funzionamento dei led dall'uso dei tasti a cui sarebbero abbinati. Per ripristinare il collegamento tra led e tasti, si può utilizzare nuovamente '**`setleds`**' con l'opzione '**`-L`**' da sola, senza altri argomenti.
`+num` `-num`	Questa modalità attiva o disattiva [*BlocNum*].
`+caps` `-caps`	Questa modalità attiva o disattiva [*Fissamaiuscole*].
`+scroll` `-scroll`	Questa modalità attiva o disattiva [*BlocScorr*].

Segue la descrizione di alcuni esempi.

- `$` **`setleds`** [*Invio*]

  Mostra la configurazione attuale.

- `$` **`setleds +num`** [*Invio*]

  Attiva i numeri nella tastiera numerica.

- `$` **`setleds -L +scroll`** [*Invio*]

  Accende il led '**`BlocScorr`**' (o '**`ScrollLock`**' che sia) senza altro effetto sull'uso della tastiera.

- `$` **`setleds -L`** [*Invio*]

Ripristina il collegamento tra led e tastiera.

- # **setleds +num < /dev/tty1** [*Invio*]

  Attiva i numeri nella tastiera numerica della prima console virtuale.

14.3.5 Mappa della tastiera

«

Il primo problema che si incontra dal punto di vista dell'internazionalizzazione di un sistema operativo, è la disposizione dei tasti sulla tastiera. Quando la tastiera viene utilizzata in modalità ASCII o Unicode, il kernel utilizza una tabella di conversione prima di trasmettere alle applicazioni i tasti premuti.

In fase di compilazione del kernel viene definita una tabella di partenza attraverso il file '***sorgenti_linux***`/driver/char/defkeymap.c`'. Normalmente questo file corrisponde alla mappa della tastiera statunitense, ma può anche essere cambiato come viene mostrato in seguito.

Di solito, la mappa della tastiera viene ridefinita attraverso il programma '**`loadkeys`**' indicando come argomento il nome di un file contenente la mappa desiderata. I file delle varie mappe disponibili sono contenuti normalmente a partire dalla directory '`/usr/share/keymaps/`'.

Il sistema della mappa della tastiera che si descrive qui e nelle sezioni seguenti, riguarda solo le console virtuali di GNU/Linux e non l'impostazione fatta dal sistema grafico X per i programmi che si avviano al suo interno.

14.3.6 Visualizzazione dei codici dei tasti con «showkey»

Il programma '**showkey**'[3] permette di visualizzare, attraverso lo standard output, il codice corrispondente ai tasti che si premono e si rilasciano. Dal momento che ciò impegna totalmente la tastiera, '**showkey**' conclude il suo funzionamento dopo 10 s di inattività.

```
showkey [opzioni]
```

Questo programma non può funzionare in una finestra di terminale nel sistema grafico X.

Tabella 14.30. Alcune opzioni.

Opzione	Descrizione
`-s` `--scancodes`	Fa in modo che siano mostrati i codici *scancode*. L'utilizzo della tastiera in questa modalità è abbastanza inusuale, quindi è raro che possa essere utile conoscere la corrispondenza della pressione e rilascio dei tasti in tale modalità.
`-k` `--keycode`	Fa in modo che siano mostrati i codici *keycode*. È il funzionamento predefinito, quando non si indicano argomenti di alcun genere. È questa la codifica a cui si fa riferimento quando si costruiscono i file di mappa gestiti da '**loadkeys**'.
`-m` `--keymap`	Mostra la traduzione dei tasti premuti nel codice corrispondente della tastiera virtuale, secondo la configurazione attiva. Si usa normalmente per verificare il funzionamento corretto di una mappa.

Segue la descrizione di alcuni esempi.

- $ **showkey -s** [*Invio*]

```
kb mode was XLATE

press any key (program terminates after 10s of last
keypress)...
```

[*Esc*]

```
0x01 0x81
```

[*Alt*]

```
0x38 0xb8
```

[*AltGr*]

```
0xe0 0x38 0xe0 0xb8
```

- $ **showkey -k** [*Invio*]

```
kb mode was XLATE

press any key (program terminates after 10s of last
keypress)...
```

[*Esc*]

```
0x01 0x81
```

[*Alt*]

```
0x38 0xb8
```

[*AltGr*]

```
0x64 0xe4
```

- $ **showkey -m** [*Invio*]

```
kb mode was XLATE

press any key (program terminates after 10s of last
keypress)...
```

[*Esc*]

```
keycode  27 press
```

[*F1*]

```
keycode  27 press
keycode  91 press
keycode  91 press
keycode  65 press
```

[*F2*]

```
keycode  27 press
keycode  91 press
keycode  91 press
keycode  66 press
```

14.3.7 Caricamento di una mappa con «loadkeys»

Il programma '**`loadkeys`**'[4] viene usato normalmente per cambiare la mappa della tastiera utilizzata in tutte le console virtuali, attraverso le indicazioni contenute in un file fornito come argomento o attraverso lo standard input. Il file fornito come argomento, se non contiene l'indicazione di un percorso, viene cercato a partire dalla directory '`/usr/share/keymaps/`***piattaforma***`/`'.

`loadkeys` [***opzioni***] [***file***]

È importante considerare che la modifica interviene su tutte le console virtuali; pertanto, se si tenta qualcosa del genere attraverso una connessione remota si interviene sull'elaboratore con il quale si è connessi e non su quello dal quale si sta operando.

Il programma '**loadkeys**' può essere utilizzato anche solo per generare un file sorgente da utilizzare al posto di '***sorgenti_linux***`/drivers/char/defkeymap.c`' quando si compila un nuovo kernel.

Tabella 14.43. Alcune opzioni.

Opzione	Descrizione
`-m` `--mktable`	Emette attraverso lo standard output il contenuto di un file che può essere utilizzato al posto di '***sorgenti_linux***`/drivers/char/defkeymap.c`' in modo da essere incorporato nel kernel alla prossima compilazione.
`-u` `--unicode`	Imposta la modalità Unicode.

Segue la descrizione di alcuni esempi.

- `$` **`loadkeys /usr/share/keymaps/i386/qwerty/it.kmap`** [*Invio*]

  Carica la mappa contenuta nel file '`/usr/share/keymaps/i386/qwerty/it.kmap`'.

- `$` **`loadkeys it.kmap`** [*Invio*]

  Esattamente come nell'esempio precedente, supponendo di operare su una piattaforma x86.

- `$` **`loadkeys it`** [*Invio*]

Esattamente come nell'esempio precedente.

- `$` **`loadkeys -u it`**[*Invio*]

  Come nell'esempio precedente, usando però la modalità Unicode.

- `$` **`loadkeys -m it > drivers/char/defkeymap.c`**[*Invio*]

  Genera il file '`drivers/char/defkeymap.c`' in base alla mappa '**`it.kmap`**'.

14.3.8 Analisi della configurazione attuale con «dumpkeys»

Il programma '**`dumpkeys`**'[5] viene usato normalmente per emettere attraverso lo standard output la mappa attuale della tastiera.

```
dumpkeys [opzioni]
```

Attraverso le opzioni si può controllare la quantità delle informazioni ottenute e il modo in cui queste vengono visualizzate.

Tabella 14.44. Alcune opzioni.

Opzione	Descrizione
`-i` `--short-info`	Emette alcune informazioni essenziali sulla gestione attuale della tastiera.
`-l` `--long-info`	Emette le informazioni relative all'opzione '**`-i`**', assieme ai simboli che possono essere utilizzati.

Opzione	Descrizione
`-h` `--help`	Emette una piccola guida di utilizzo del programma, aggiungendo in coda l'elenco delle codifiche che si possono usare per la configurazione della tastiera. L'elenco delle codifiche si ottiene precisamente dallo standard error.

Segue la descrizione di alcuni esempi.

- `$` **`dumpkeys -i`** [*Invio*]

```
keycode range supported by kernel:           1 - 255
max number of actions bindable to a key:         256
number of keymaps in actual use:                 8
of which 2 dynamically allocated
ranges of action codes supported by kernel:
        0x0000 - 0x00ff
        0x0100 - 0x01ff
        0x0200 - 0x0213
        0x0300 - 0x0313
        0x0400 - 0x0405
        0x0500 - 0x05ff
        0x0600 - 0x0603
        0x0700 - 0x0708
        0x0800 - 0x08ff
        0x0900 - 0x0919
        0x0a00 - 0x0a07
        0x0b00 - 0x0bff
        0x0c00 - 0x0c07
        0x0d00 - 0x0dff
number of function keys supported by kernel: 256
max nr of compose definitions: 256
nr of compose definitions in actual use: 68
```

- $ **dumpkeys -l** [*Invio*]

```
…
Symbols recognized by dumpkeys:
(numeric value, symbol)

0x0000  nul
0x0001  Control_a
0x0002  Control_b
…
0x00fe  thorn
0x00ff  ydiaeresis
0x0100  F1
0x0101  F2
…
0x0112  F19
0x0113  F20
0x0114  Find
0x0115  Insert
0x0116  Remove
…
0x01fe  F245
0x01ff  F246
0x0200  VoidSymbol
0x0201  Return
…
0x0404  dead_diaeresis
0x0405  dead_cedilla
0x0500  Console_1
0x0501  Console_2
…
0x0c06  SCtrlL
0x0c07  SCtrlR

The following synonyms are recognized:
```

```
Control_h       for BackSpace
Control_i       for Tab
…
Recognized modifier names and their column numbers:
shift             1
altgr             2
control           4
alt               8
shiftl           16
shiftr           32
ctrll            64
ctrlr           128
```

- `$` **`dumpkeys > prova.map`**[*Invio*]

  Crea il file ‘`prova.map`’ contenente la configurazione attuale della tastiera. Il file potrebbe avere l’aspetto seguente:

  `$` **`cat prova.map`**[*Invio*]

```
keymaps 0-2,4,6,8-9,12
keycode   1 = Escape            Escape
        alt     keycode   1 = Meta_Escape
keycode   2 = one               exclam
        alt     keycode   2 = Meta_one
        shift   alt     keycode   2 = Meta_exclam
…
keycode  83 = KP_Period
        altgr   control keycode  83 = Boot
        control alt     keycode  83 = Boot
…
string F1 = "\033[[A"
string F2 = "\033[[B"
string F3 = "\033[[C"
…
compose 't' 'h' to 'þ'
```

```
compose 's' 's' to 'ß'
compose '"' 'y' to 'ÿ'
compose 's' 'z' to 'ß'
compose 'i' 'j' to 'ÿ'
```

- `$` **`dumpkeys | loadkeys --unicode`** [*Invio*]

  Riconfigura la tastiera in modo da emettere simboli secondo la codifica UTF-8.

- `$` **`dumpkeys -h > /dev/null`** [*Invio*]

  Visualizza esclusivamente lo standard error, dal quale si ottiene l'elenco delle codifiche disponibili:

```
iso-8859-{1,2,3,4,5,7,8,9,10,15},koi8-{r,u},mazovia,↩
↪cp-1250,iso-10646-18,iso-ir-197,iso-ir-209
```

14.4 Personalizzazione della mappa della tastiera

Per configurare la propria tastiera, in un sistema GNU/Linux, si usa generalmente '**`loadkeys`**' con file di mappa già predisposti; tuttavia può essere utile comprendere come si realizzano questi file per una propria personalizzazione.

All'interno del file sono ammessi i commenti, prefissati con un punto esclamativo ('**`!`**') oppure con il simbolo '**`#`**'; nello stesso modo sono ignorate le righe vuote e quelle bianche. Le righe che definiscono qualcosa possono essere continuate con una barra obliqua inversa ('**`\`**') che precede il codice di interruzione di riga.

Possono essere usate diverse definizioni; in particolare quelle descritte nella tabella seguente.

Tabella 14.49. Direttive principali.

Direttiva	Descrizione
`charset` ***"codifica"***	Questa direttiva definisce l'insieme di caratteri utilizzato; generalmente si tratta di '**`iso-8859-1`**' o di '**`iso-10646-18`**' (nel secondo caso si fa riferimento a Unicode). Quando si tratta della codifica ISO 8859-1, oppure ISO 10646-18, non è nemmeno necessario inserire questo tipo di dichiarazione nei file. Lo scopo di questa direttiva è di evitare ambiguità, quando un carattere può essere rappresentato con codici differenti a seconda della codifica.
`keycode` ***numero_tasto*** = ***simbolo*** ***simbolo***...	Questa definizione attribuisce a un tasto diversi significati, in funzione della combinazione eventuale con altri.

Direttiva	Descrizione
`string` ***nome*** `=` ***stringa*** `strings as usual`	Per facilitare la costruzione di un file di mappa del genere, si possono definire alcuni nomi di tasti il cui significato viene chiarito in seguito attraverso questo tipo di dichiarazione. Lo si fa normalmente con i tasti funzionali ([*F1*], [*F2*], ecc.). Al posto della dichiarazione esplicita di queste stringhe, si può usare la seconda forma, che richiama automaticamente tutte quelle predefinite.
`compose` ***'x' 'y'*** `to` ***'z'*** `compose as usual for iso-8859-1`	La direttiva '**`compose`**' consente di stabilire l'uso di accenti morti: il carattere ***x***, da associare a un carattere successivo, ***y***, per generare un carattere composto, ***z***. Utilizzando la seconda forma del modello, si acquisiscono tutte le sequenze predefinite.

A proposito della direttiva '**charset**' è bene chiarire che il codice associato ai caratteri è da intendersi come punto di codifica; pertanto, per fare un esempio, non c'è differenza tra il codice generato da una lettera «à» (a minuscola con accento grave) da una codifica ISO 8859-1 a una codifica ISO 10646-18, perché in questa fase non entra in gioco il fatto che nell'adattamento in UTF-8 si utilizzano due byte invece di uno solo.

14.4.1 Modificatori

«

I modificatori sono quei tasti che si possono usare per ottenere delle combinazioni. La tabella 14.50 ne mostra l'elenco e il ***peso***.

Tabella 14.50. Elenco dei tasti modificatori e del loro peso.

Modificatore	Sigla	peso
(nessuno)		0
Maiuscole (tasto sinistro o destro indifferentemente)	`Shift`	1
Alt destro	`AltGr`	2
Control (tasto sinistro o destro indifferentemente)	`Ctrl`	4
Alt sinistro	`Alt`	8
Maiuscole sinistro	`ShiftL`	16
Maiuscole destro	`ShiftR`	32
Control sinistro	`CtrlL`	64
Control destro	`CtrlR`	128

I modificatori sono otto, a cui si somma la situazione normale in cui nessun modificatore viene utilizzato. Volendo indicare tutte le combinazioni possibili di modificatori, queste sarebbero 255, ma di solito ci si limita a configurarne solo un sottoinsieme.

Attraverso il numero del peso, si può fare riferimento a un modificatore o a una combinazione di modificatori, in modo molto semplice: sommandone i valori. Per esempio, uno rappresenta '**Shift**', due rappresenta '**AltGr**' e tre rappresenta '**Shift**'+'**AltGr**'. Si osservi comunque che si sta facendo riferimento a nomi che poi devono essere associati alla tastiera reale: si presume che l'associazione avvenga nel modo «corretto», ma occorre sapere a cosa si sta facendo riferimento. In pratica, nulla vieta di dare la funzione di '**AltGr**' a un tasto [*Win*], o [*Menù*].

14.4.2 Specificazione di mappa

Quando si specifica la funzione di un tasto attraverso l'istruzione '**keycode**', si indicano delle funzioni in sequenza. Il significato di questa sequenza dipende dai tipi di modificatori e dalle loro combinazioni che si intendono utilizzare. Ciò viene definito attraverso un'istruzione '**keymaps**' iniziale.

```
keymaps peso [,peso] ...
```

Per esempio, l'istruzione seguente indica l'utilizzo dei pesi 0, 1, 2, 4, 6, 8, 9 e 12:

```
keymaps 0-2,4,6,8-9,12
```

Come si vede nell'esempio, si fa riferimento anche a intervalli, quindi, '**0-2**' rappresenta tutti i valori da zero a due, ovvero, zero, uno e

due. La stessa cosa avrebbe potuto essere dichiarata in un modo più esplicito come nell'esempio seguente:

```
keymaps 0,1,2,4,6,8,9,12
```

Questi valori indicano che nella mappa definita dalle direttive successive, si fa riferimento ai tasti premuti da soli, in combinazione con i modificatori '**Shift**', '**AltGr**', '**Ctrl**' (indifferentemente sinistro o destro), '**Ctrl**'+'**AltGr**', '**Alt**', '**Alt**'+'**Shift**' e '**Ctrl**'+'**Alt**'. Le istruzioni '**keycode**' successive all'istruzione '**keymaps**' dell'esempio vengono interpretate di conseguenza. L'esempio seguente dovrebbe chiarirlo:

```
keycode 26 = egrave  eacute  bracketleft  Escape  VoidSymbol  Meta_bracketleft
```

In questo caso, premendo il tasto corrispondente alla lettera '**è**', nella tastiera italiana, si ottiene esattamente questa lettera ('**egrave**'). In combinazione con:

- '**Shift**' si ottiene la lettera '**é**';
- '**AltGr**' si ottiene la parentesi quadra aperta (sinistra);
- '**Ctrl**' si ottiene un escape;
- '**Ctrl**'+'**AltGr**' non si ottiene alcunché;
- '**Alt**' si ottiene la funzione <*Meta_braketleft*>.

In tutti i casi rimanenti non si ottiene alcun risultato.

14.4.3 Istruzione «keycode»

«

L'istruzione '**keycode**' permette di indicare in sequenza il significato di un certo tasto, in funzione dell'eventuale combinazione con i modificatori previsti con l'istruzione '**keymaps**'.

Quando si vuole indicare un'azione nulla, si usa il nome '**VoidSymbol**'; inoltre, ciò che non viene scritto nella parte finale vale come se fosse sempre '**VoidSymbol**'.

Per facilitare l'indicazione del risultato di combinazioni si possono usare dichiarazioni '**keycode**' successive, precedute dai nomi dei modificatori coinvolti, le quali valgono per una singola situazione. L'esempio seguente è identico, per risultato, a quello visto in precedenza; la differenza sta nel fatto che così ci si limita a indicare un'istruzione '**keycode**' normale per le situazioni comuni (tasto premuto da solo, oppure in combinazione con il modificatore '**Shift**', o anche '**AltGr**'), aggiungendo sotto, eventualmente, le altre combinazioni utili.

```
keymaps 0,1,2,4,6,8,9,12
…
keycode  26 = egrave            eacute            bracketleft
        control keycode  26 = Escape
        alt     keycode  26 = Meta_bracketleft
```

Volendo essere precisi e dettagliati, si può indicare il modificatore '**plain**' per fare riferimento alla pressione del tasto senza modificatori. L'esempio appena mostrato si traduce in quello seguente:

```
keymaps 0,1,2,4,6,8,9,12
…
plain   keycode 26 = egrave
shift   keycode 26 = eacute
altgr   keycode 26 = braceleft
control keycode 26 = Escape
alt     keycode 26 = Meta_bracketleft
```

Quando si usa la direttiva '**keycode**' si fa riferimento ai caratteri e

alle funzioni speciali attraverso dei nomi. Tali nomi possono corrispondere, eccezionalmente, alla lettera che rappresenta un carattere, quando questa appartiene all'alfabeto inglese, senza accenti. I nomi disponibili si possono consultare nell'elenco che si ottiene con il programma '**dumpkeys**', utilizzato con l'opzione '**-l**':

```
$ dumpkeys -l [Invio]

  …
  Symbols recognized by dumpkeys:
  (numeric value, symbol)

  0x0000  nul
  0x0001  Control_a
  0x0002  Control_b
  …
  0x00fe  thorn
  0x00ff  ydiaeresis
  0x0100  F1
  0x0101  F2
  …
  0x0112  F19
  0x0113  F20
  0x0114  Find
  0x0115  Insert
  0x0116  Remove
  …
  0x01fe  F245
  0x01ff  F246
  0x0200  VoidSymbol
  0x0201  Return
  …
  0x0404  dead_diaeresis
  0x0405  dead_cedilla
  0x0500  Console_1
```

```
0x0501  Console_2
…
0x0c06  SCtrlL
0x0c07  SCtrlR

The following synonyms are recognized:

Control_h       for BackSpace
Control_i       for Tab
…
Recognized modifier names and their column numbers:
shift             1
altgr             2
control           4
alt               8
shiftl           16
shiftr           32
ctrll            64
ctrlr           128
```

Quando si tratta di caratteri tipografici particolari, può capitare che non siano disponibili dei nomi per individuarli. In quel caso, si può indicare il punto di codifica Unicode, secondo la notazione seguente:

```
U+hhhh
```

Per esempio, ecco come potrebbe apparire la definizione relativa al tasto [*o*], quando non si sa che la lettera «ø» si identifica con il nome '**oslash**' e la maiuscola con '**Oslash**':

```
plain       keycode 24 = o
shift       keycode 24 = O
altgr       keycode 24 = U+00F8
shift altgr keycode 24 = U+00D8
```

In pratica, con il modificatore '**AltGr**' si intende ottenere il simbolo «ø», ma non sapendo come fare, si utilizza la notazione U+00F6, così come per la maiuscola, «Ø», si utilizza la notazione U+00D8.

La notazione attraverso l'indicazione del punto di codifica va bene sempre, anche se si usa una codifica del tipo ISO 8859 (a soli 8 bit).

14.4.4 Funzionalità speciali

«

Studiando un file di mappa della tastiera si possono trovare alcune cose interessanti, come la definizione di combinazioni particolari. Gli estratti riportati di seguito provengono dalla mappa italiana normale: '`/usr/share/keymaps/i386/qwerty/it.kmap`'. I modificatori utilizzati sono quelli degli esempi precedenti, ovvero: 0, 1, 2, 4, 6, 8, 9 e 12.

```
keycode  57 = space             space
        control keycode  57 = nul
        alt     keycode  57 = Meta_space
        control alt keycode 57 = Meta_nul
```

Nell'esempio appena mostrato si nota che la combinazione [*Ctrl Spazio*] genera un carattere <*NUL*>.

```
keycode  59 = F1                F11              Console_13
        control keycode  59 = F1
        alt     keycode  59 = Console_1
        control alt     keycode  59 = Console_1
keycode  60 = F2                F12              Console_14
        control keycode  60 = F2
        alt     keycode  60 = Console_2
        control alt     keycode  60 = Console_2
…
string F1 = "\033[[A"
string F2 = "\033[[B"
…
string F11 = "\033[23~"
string F12 = "\033[24~"
…
```

Quello che appare sopra è la dichiarazione del comportamento dei tasti funzionali. I nomi di questi tasti non sono riconosciuti e quindi si dichiara più avanti la stringa che deve essere generata quando si fa riferimento a questi.

Si può osservare che la combinazione [*Maiuscole F1*] genera l'equivalente di [*F11*].

La combinazione [*Alt F1*] o [*Ctrl Alt F1*] serve notoriamente per selezionare la prima console virtuale, cosa che viene definita chiaramente con le istruzioni '**`alt keycode 59 = Console_1`**' e '**`control alt keycode 59 = Console_1`**'. Nello stesso modo si può osservare che la combinazione [*AltGr F1*] seleziona la tredicesima console virtuale (ammesso che ci sia).

```
keycode  70 = Scroll_Lock       Show_Memory      Show_Registers
        control keycode  70 = Show_State
        alt     keycode  70 = Scroll_Lock
```

Da questa dichiarazione, si osserva che la combinazione [*Maiuscole BlocScorr*] visualizza la situazione dell'uso della memoria, la combinazione [*AltGr BlocScorr*] mostra la situazione dei registri e la combinazione [*Ctrl BlocScorr*] mostra lo stato.

L'esempio seguente è tratto da una mappa per una tastiera russa, che prevede i primi due livelli con un alfabeto latino, mentre il terzo livello può essere fissato utilizzando il tasto [*Ctrl*] destro:

```
# Russian Cyrillic keyboard.map. "Cyrillic" mode is
# toggled by Right_Ctrl key and shifted by AltGr key.
keymaps 0-4,6,8,10,12
strings as usual
…
                keycode 96 =    KP_Enter
                keycode 97 =    AltGr_Lock
                keycode 98 =    KP_Divide
…
```

Si può osservare l'uso della definizione '**`AltGr_Lock`**' per ottenere il passaggio stabile al terzo livello, come se qualcuno tenesse sempre premuto il tasto [*AltGr*]. Naturalmente, per sbloccare la selezione del terzo livello basta ripremere lo stesso tasto già usato per inserirlo.

14.5 Approfondimento: tastiera italiana conforme a X

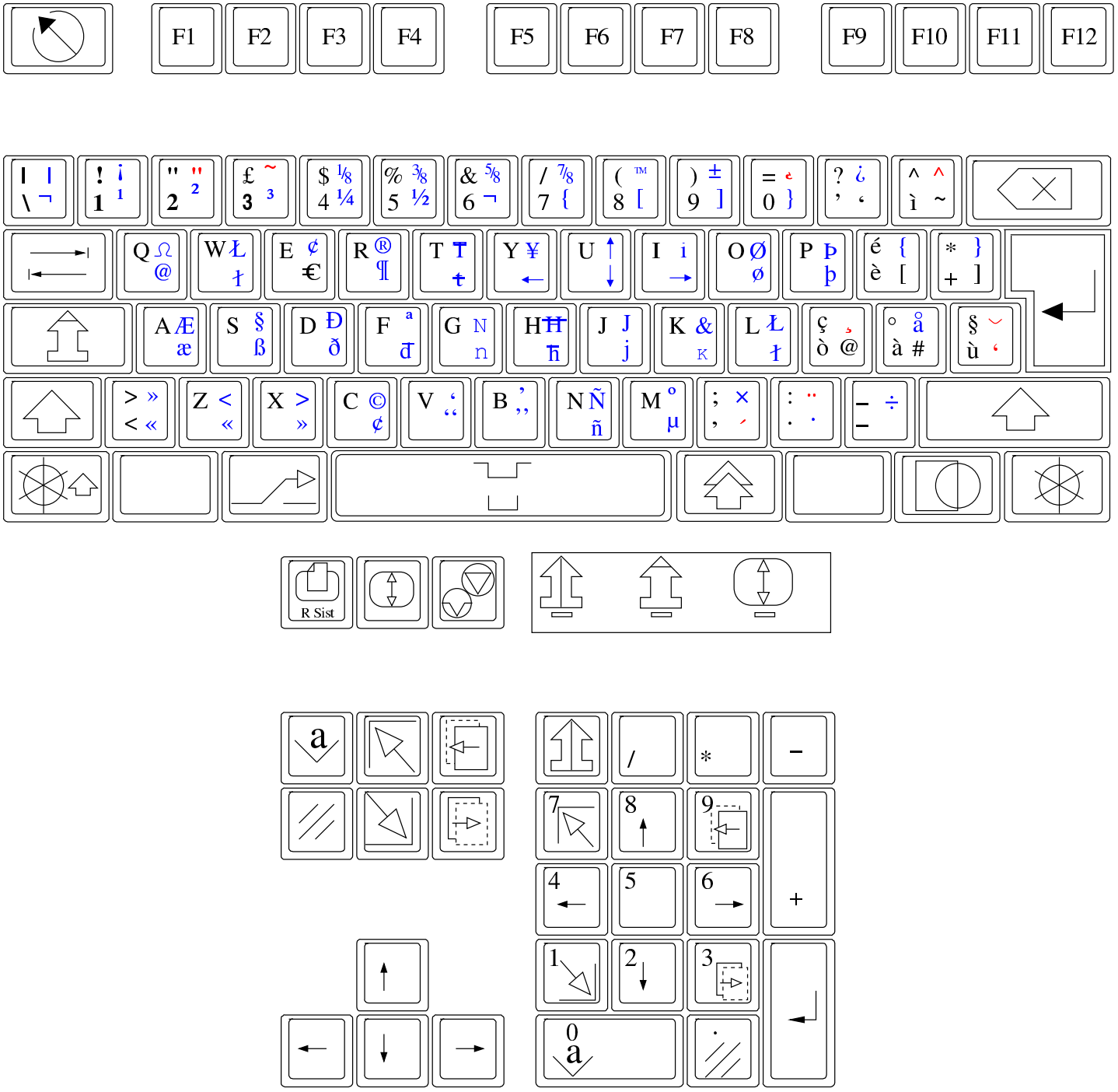
«

Esistono diverse versioni di mappe per la tastiera italiana, diffuse comunemente con le distribuzioni GNU/Linux. Dovrebbe trattarsi dei file '`it.kmap.gz`', '`it2.kmap.gz`' e '`it-ibm.kmap.gz`', contenuti normalmente nella directory '`/usr/share/keymaps/i386/qwerty/`'. Nessuna di queste mappe soddisfa l'esigenza di buona compatibilità con quella usata con X: simboli importanti come la

tilde, le parentesi graffe e le virgolette basse si trovano spesso in posizioni differenti. Oltre a questo, la mappa di X offre delle alternative nei livelli tre e quattro, provenienti dallo standard ISO 9995 ed è un peccato non poter avere un modo uniforme per ottenere i simboli desiderati.

Questa sezione descrive la configurazione di una mappa italiana, estesa, con l'obiettivo di essere il più possibile conforme al modello di quella usata con X, come si vede dalla figura 14.62. Il sorgente della mappa che viene descritta è disponibile presso *allegati/it-xorg.kmap* . Si osservi che non è prevista la presenza di un tasto per fissare il terzo livello ('**`AltGr_Lock`**'), perché la mappa italiana consente normalmente di ottenere tutti i simboli necessari per scrivere in lingua inglese e il passaggio al terzo livello viene richiesto raramente (per esempio per la chiocciola, le parentesi quadre e quelle graffe).

Figura 14.62. Tastiera italiana conforme a X, divisa in due parti per motivi di spazio. I simboli di colore diverso dal nero sono estensioni alla tastiera italiana; i simboli disegnati sul lato destro dei tasti si ottengono con il terzo e quarto livello; i simboli in rosso sono accenti morti; le lettere accentate maiuscole si ottengono premendo anche il tasto [*Ctrl*] sinistro.



14.5.1 Codifica

La configurazione della tastiera proposta non riporta l'indicazione della codifica, perché può essere indifferentemente ISO 8859-1 o ISO 10646-18. Naturalmente, quando si utilizza '**`loadkeys`**' con questa mappa e la console non sta funzionando secondo la codifica UTF-8, si ottengono dei messaggi di errore, non fatali, per i caratteri che con ISO 8859-1 non possono essere ottenuti. Ovviamente, tali caratteri possono essere ottenuti solo quando la console può gestirli.

14.5.2 Modificatori virtuali «control» e «meta»

Il modificatore virtuale '**`Control`**' è abbinato indifferente a uno dei due tasti [*Ctrl*], mentre il modificatore virtuale '**`Meta`**' è abbinato al tasto [*Alt*] sinistro.

Le combinazioni virtuali di questi modificatori sono associate in modo da poter essere ottenute secondo un criterio logico, anche se non è detto che siano facili da produrre. Per esempio, la parentesi quadra chiusa appare sia nel tasto utilizzato normalmente per ottenere la lettera «è» (e con accento grave), sia nel tasto utilizzato normalmente per ottenere il numero otto. Pertanto, la combinazione virtuale <*Meta_bracketleft*> si ottiene come [*Alt [*], che in pratica può essere ottenuta come [*Alt AltGr è*], oppure [*Alt AltGr 8*]. Naturalmente, quando il simbolo da combinare si trova nel quarto livello, occorre inserire nella combinazione reale anche il tasto [*Maiuscole*].

Si osservi che la combinazione reale [*Maiuscole Esc*] genera il carattere <*NUL*> (pari allo zero binario), pertanto la combinazione virtuale <*Meta_nul*> si ottiene come [*Maiuscole Alt Esc*].

14.5.3 Combinazioni numeriche

«

Tenendo premuto il tasto [*Alt*], oppure il tasto [*Ctrl*], è possibile comporre sulla tastiera numerica il numero (in base dieci) del carattere che si vuole ottenere. Il carattere si ottiene quando si rilascia il tasto usato per iniziare la sequenza ([*Alt*] o [*Ctrl*]). Si può fare la stessa cosa usando i numeri che appaiono sulla parte superiore della parte alfanumerica della tastiera, ma in tal caso si può usare soltanto il tasto [*Ctrl*].

Tenendo premuta la combinazione [*Maiuscole Ctrl*], è possibile comporre il numero di un carattere che si vuole ottenere, ma in esadecimale. Per le cifre numeriche da uno a nove si può usare indifferentemente la tastiera numerica o quella alfanumerica.

14.5.4 Accenti morti e composizione

«

Le sequenze con cui si generano le lettere accentate a partire dagli accenti morti sono quelle predefinite; pertanto possono essere differenti rispetto a quelle usate nella configurazione di X. Oltre a questo fatto, è da notare che non è possibile definire un accento morto del tipo *dead ring*, pertanto, con la combinazione reale [*Maiuscole AltGr à*] si ottiene direttamente la lettera «å», che è l'unica prevista per quella ipotetica trasformazione. Inoltre, non è possibile dichiarare un accento morto del tipo *dead macron*, pertanto in corrispondenza della combinazione reale [*AltGr −*] non si ottiene alcunché.

È previsto un tasto per iniziare una sequenza di composizione. Si tratta precisamente del tasto [*Menù*], che è disponibile solo con una tastiera a 105 tasti. Anche le sequenze di composizione sono quelle predefinite.

14.5.5 Lettere accentate maiuscole

I quattro livelli previsti, che si ottengono come [***x***], [*Maiuscole* ***x***], [*AltGr* ***x***] e [*Maiuscole AltGr* ***x***], non consentono di produrre alcune lettere accentate. X produce tali lettere utilizzando il tasto [*Fissamaiuscole*], ma nella console questo non è distinguibile da un tasto [*Maiuscole*] che rimane premuto. Pertanto, è stato necessario attribuire tale compito ai tasti [*Ctrl*], così, per ottenere la lettera «È» (e maiuscola con accento grave), si può usare la combinazione [*Ctrl è*]; nello stesso modo, per ottenere la lettera «É», si può usare la combinazione [*Maiuscole Ctrl è*]. Lo stesso ragionamento vale anche quando la lettera da rendere in maiuscolo si trova a un livello superiore, come nel caso di «Å», che si può ottenere come [*Maiuscole Ctrl AltGr à*].

Si osservi che dove ciò non è necessario, la combinazione con il tasto [*Ctrl*] non è stata prevista, oppure serve per generare una combinazione virtuale del tipo <*Control_**x***>.

14.5.6 Barra spaziatrice

La pressione della barra spaziatrice produce normalmente il carattere <*SP*>, ma se si usa in combinazione con il tasto [*Ctrl*], si ottiene uno spazio non interrompibile (o non separabile), che corrisponde al codice U+00A0.

14.5.7 Tasti funzionali

I tasti funzionali [*F1*], [*F2*],... [*F12*], agiscono come di consueto, se però si aggiunge in combinazione il tasto [*Maiuscole*], ci si sposta in avanti di 10 posizioni: [*Maiuscole F1*] equivale a [*F11*], [*Maiuscole F2*] equivale a [*F12*],...

La selezione delle console virtuali avviene come di consueto, utilizzando la combinazione [*Alt Fn*], oppure [*Ctrl Alt Fn*]; se si aggiunge nella combinazione il tasto [*Maiuscole*], si ottiene la console ***n***+10.

14.6 Identificazione del terminali

«

Il terminale, in qualunque forma esso sia (console, terminale remoto, applicazione a finestra all'interno di X) è il mezzo normale di comunicazione tra l'utente e il sistema. Senza di esso non ci sarebbe alcuna possibilità di avviare nuovi processi e, di conseguenza, nemmeno di poter compiere alcuna attività.

Per questo, l'attivazione di un programma per la gestione del terminale è l'ultima fase di una procedura di inizializzazione del sistema e precede immediatamente l'attivazione della procedura di accesso (il *login*), cioè il sistema di riconoscimento dell'utente che si accinge a utilizzare il sistema operativo. I programmi Getty che sono i responsabili dell'attivazione del terminale prima dell'inizio della procedura di accesso, sono introdotti nella sezione 14.15.

È importante poter identificare il terminale da cui si accede, almeno in base al tipo di dispositivo utilizzato. In pratica, si dispone del programma '**`tty`**'[6] che è in grado di restituire il nome del file di dispositivo corrispondente. Con questa informazione si possono creare degli script opportuni, eventualmente per filtrare l'accesso da parte degli utenti.

`tty` [*opzioni*]

Il programma '**tty**' emette attraverso lo standard output il nome del terminale con cui si è connessi.

Tabella 14.63. Alcune opzioni.

Opzione	Descrizione

`-s`
`--silent`
`--quiet`
	Non emette alcuna segnalazione, si limita a restituire un valore.

L'esempio seguente mostra in che modo potrebbe essere utile '**tty**'. Se l'utente sta utilizzando la prima console virtuale ('`/dev/tty1`'), viene respinto; altrimenti viene eseguito il comando '**ls**'.

```
#!/bin/sh
if [ `tty` = "/dev/tty1" || `tty` = "/dev/vc/1" ]
then
    echo "spiacente, non puoi usare questo terminale"
else
    ls
fi
```

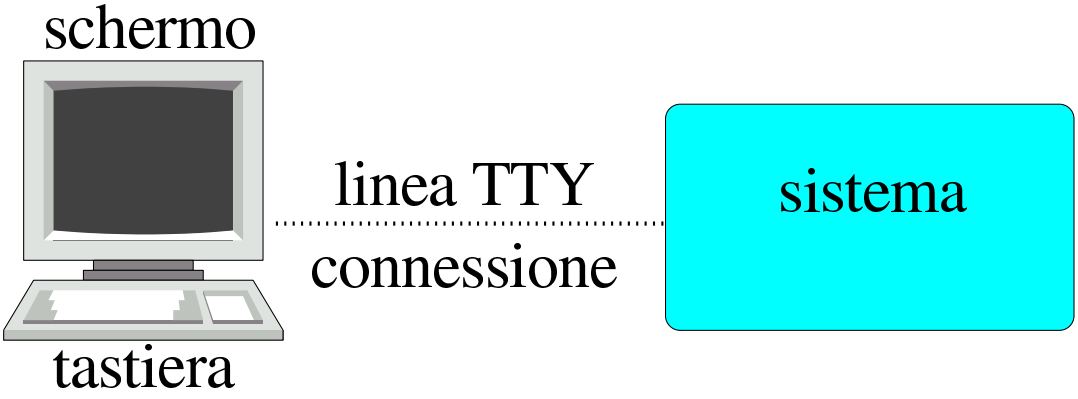
14.7 Configurazione del terminale

Le caratteristiche dei terminali a caratteri possono essere molto diverse e questo è il problema principale che si pone di fronte alla ricerca verso una standardizzazione nel comportamento dei programmi per i sistemi Unix.

Si distinguono due problemi di ordine diverso: la configurazione del I/O (input-output), ovvero dei flussi di dati tra il terminale (TTY) e il sistema, e la configurazione particolare dello schermo. La con-

figurazione dei flussi di dati regola il modo in cui i dati possono essere inseriti attraverso la tastiera e come questo inserimento può essere controllato sullo schermo durante la sua digitazione (eco); la configurazione dello schermo riguarda il modo di rappresentare simboli determinati e di comportarsi di fronte a sequenze di escape determinate.

Figura 14.65. Schema banale della connessione di un terminale a caratteri.



Il tipo di connessione utilizzata (si pensi alla differenza che c'è tra una console legata strettamente con il sistema, rispetto a un terminale seriale o remoto), implica problemi differenti di gestione della linea TTY. L'utilizzatore normale non ha mai bisogno di preoccuparsi di questo, in quanto per ogni situazione c'è già un'impostazione predefinita che dovrebbe soddisfare le esigenze di tutti. Inoltre, nelle connessioni remote, il problema di questa configurazione si sposta sui programmi che si utilizzano per tali scopi; sono poi questi programmi a definire la configurazione della linea e dei flussi di dati elementari.

All'utente è data la possibilità di verificare questa configurazione e di modificarla, attraverso il programma '**`stty`**' (*Set tty*).

La fase successiva è la definizione delle particolarità degli scher-

mi dei terminali, per ciò che riguarda le sequenze di escape che questi riconoscono, attraverso una sorta di base di dati, in modo da permettere ai programmi di potervisi adattare.

14.7.1 Linea TTY

Prima di descrivere l'utilizzo sommario di '**stty**', conviene prendere confidenza con il problema, attraverso un po' di esercizio.

```
$ cat > /dev/null [Invio]
```

Avviando il programma '**cat**' in questo modo, si può analizzare ciò che succede quando si inserisce qualcosa attraverso la tastiera del proprio terminale.

```
asdfghjkl [Invio]

qwertyuiop [Invio]
```

Digitando lettere normali, queste appaiono semplicemente sullo schermo. L'eco dell'input, non è una cosa scontata; deriva da una configurazione, anche se questa è generalmente predefinita.

```
[Ctrl p][Ctrl l][Esc][F1][F2][Invio]

  ^P^L^[^[[A^[[B
```

Generalmente, i caratteri di controllo che non hanno significati speciali, vengono visualizzati (eco) come lettere maiuscole (o brevi stringhe) precedute da un accento circonflesso, come mostra l'esempio. Si tratta di una caratteristica configurabile, anche se normalmente è già impostata in questo modo.

Ad alcuni caratteri di controllo viene attribuito un significato speciale, il quale si traduce in un comportamento e non nell'eco di un

qualche simbolo.

asdf ghjk lqwe rtyu iop [*Ctrl ?*][*Ctrl ?*][*Ctrl ?*][*Ctrl w*][*Invio*]

```
asdf ghjk lqwe
```

La combinazione [*Ctrl ?*] genera normalmente il carattere speciale <^*?*>, il quale di solito è abbinato alla funzione '**erase**', la quale a sua volta si traduce nella cancellazione dell'ultimo carattere inserito. La combinazione [*Ctrl w*] genera normalmente il carattere speciale <^*W*>, il quale di solito è abbinato alla funzione '**werase**', la quale a sua volta si traduce nella cancellazione dell'ultima parola inserita.

Ad altri caratteri di controllo viene abbinato l'invio di un segnale al processo collegato alla linea di terminale. Ecco che così, di solito, la combinazione [*Ctrl c*] genera il carattere speciale <^*C*>, con il quale viene inviato un segnale '**SIGINT**' al processo collegato. Nello stesso modo, la combinazione [*Ctrl z*] genera il carattere speciale <^*Z*>, con il quale viene inviato un segnale '**SIGTSTP**' al processo collegato (cosa che generalmente si traduce nell'essere messo sullo sfondo dalla shell).

Per concludere questo esercizio, basta utilizzare la combinazione [*Ctrl c*], per terminare il funzionamento di '**cat**'.

[*Ctrl c*]

Un'altra cosa interessante è la possibilità di bloccare il flusso dell'output sullo schermo e di riprenderlo successivamente. Per questo si usano normalmente le combinazioni di tasti [*Ctrl s*] e [*Ctrl q*], le quali generano rispettivamente i codici <^*S*> e <^*Q*>.

Per verificarne il funzionamento, basta provare a lanciare un comando che emette un output molto lungo, come il seguente:

```
$ find / -print [Invio]
```

Per sospendere il flusso visualizzato sullo schermo del terminale, basta premere [*Ctrl s*]; per farlo riprendere, [*Ctrl q*].

14.7.2 Utilizzo di «stty»

Il programma '**stty**'[7] permette di modificare le caratteristiche della connessione del terminale al sistema. Se viene avviato senza argomenti, visualizza le informazioni salienti della connessione. Gli argomenti della configurazione sono delle parole chiave che possono apparire precedute o meno dal trattino che di solito si usa per le opzioni: se non si usa il trattino, la parola chiave viene intesa come attivazione di qualcosa, con il trattino si intende la disattivazione della stessa cosa.

`stty` [*opzioni* | *configurazione*]

Il motivo più comune per servirsi di questo programma è quello di conoscere le combinazioni di tasti che si possono utilizzare per generare dei segnali particolari.

Sia chiaro che i «caratteri» del tipo <^*?*>, <^*W*>, <^*C*>, <^*Z*>,... si ottengono attraverso «combinazioni virtuali»; pertanto, occorre accertarsi che la configurazione della tastiera corrisponda effettivamente, oppure occorre sapere in che modo vanno generati questi simboli nell'ambito del proprio contesto. Negli esempi che si vedono qui si suppone che il tasto [*Ctrl*] corrisponda esattamente alle funzioni del modificatore virtuale '**Control**'.

Avviando '**stty**' con l'opzione '**-a**' si ottiene la configurazione corrente.

```
$ stty -a [Invio]
```

Per esempio, si potrebbe ottenere qualcosa di simile al listato seguente:

```
speed 38400 baud; rows 25; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U;
eof = ^D; eol = <undef>; eol2 = <undef>; start = ^Q;
stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr
-igncr icrnl ixon ixoff -iuclc -ixany -imaxbel opost -olcuc
-ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0
vt0 ff0 isig icanon -iexten echo echoe echok -echonl -noflsh
-xcase -tostop -echoprt -echoctl echoke
```

L'esempio indica in particolare che il carattere '**intr**' (*interrupt*) viene generato con la combinazione [*Ctrl c*]; il carattere '**quit**' viene generato con la combinazione [*Ctrl* \]; il codice di EOF (*End of file*) viene generato con la combinazione [*Ctrl d*]; il carattere '**susp**' (*suspend*) viene generato con la combinazione [*Ctrl z*].

Per comprendere meglio il senso di questo programma, vale la pena di descrivere l'uso di alcune opzioni, anche se nella maggior parte dei casi, '**stty**' non viene usato per queste cose (tabella 14.69).

Tabella 14.69. Alcune opzioni.

Opzione	Descrizione
`cs8`	Definisce la dimensione dei caratteri a 8 bit.

Opzione	Descrizione
`hupcl` `-hupcl`	Attiva o disattiva l'invio di un segnale di aggancio ('**SIGHUP**') in corrispondenza della conclusione dell'attività dell'ultimo processo, cosa che chiude la connessione con il terminale.
`crtscts` `-crtscts`	Attiva o disattiva il controllo di flusso RTS/CTS. Evidentemente, questo tipo di controllo di flusso riguarda i terminali connessi attraverso la porta seriale.
`brkint` `-brkint`	Attiva o disattiva l'invio di un segnale di interruzione ('**SIGINT**') in corrispondenza dell'invio di un carattere *break*.
`istrip` `-istrip`	Attiva o disattiva l'azzeramento dell'ottavo bit dell'input.
`ixon` `-ixon`	Abilita o disabilita il controllo di flusso XON/XOFF. Dalla sua abilitazione dipende il funzionamento di caratteri speciali riferiti ai comandi di '**stop**' e '**start**' (di solito [*Ctrl s*] e [*Ctrl q*]).
`isig` `-isig`	Abilita o disabilita l'uso di caratteri speciali, corrispondenti ai comandi '**intr**' (*interrupt*), '**quit**' e '**susp**' (*suspend*), i quali di solito corrispondono a [*Ctrl c*], [*Ctrl* \] e [*Ctrl z*].
`icanon` `-icanon`	Abilita o disabilita l'uso di caratteri speciali, corrispondenti ai comandi '**erase**', '**kill**', '**werase**' e '**rprnt**', i quali di solito corrispondono a [*Ctrl ?*], [*Ctrl u*], [*Ctrl w*] e [*Ctrl r*].
`echo` `-echo`	Abilita l'eco dei caratteri inseriti. Senza l'attivazione di questa modalità, non sarebbe visibile l'input dalla tastiera.

Opzione	Descrizione

`echoctl`
`-echoctl`
`ctlecho`
`-ctlecho`
	Attiva o disattiva l'eco dei caratteri di controllo attraverso la notazione '**^*x***', dove ***x*** è una lettera che varia a seconda del carattere di controllo da visualizzare.

`sane`	Questa opzione è una scorciatoia per definirne una serie numerosa, allo stato predefinito ritenuto corretto generalmente. In pratica implica quanto segue: '**cread -ignbrk brkint -inlcr -igncr icrnl -ixoff -iuclc -ixany imaxbel opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke**'; inoltre imposta i caratteri speciali al loro valore predefinito.

I caratteri speciali abbinati a funzionalità particolari in modo predefinito, possono variare da un sistema all'altro. Per modificare l'attribuzione di un carattere speciale a una certa funzione, si utilizza la sintassi seguente:

```
stty nome_funzione carattere_speciale
```

Se al posto del simbolo del carattere speciale si utilizza la stringa '**^-**', oppure la parola chiave '**undef**', quella funzionalità viene disabilitata.

Segue l'elenco di alcune parole chiave utilizzate per definire

funzionalità a cui si possono attribuire caratteri speciali.

Tabella 14.70. Alcuni caratteri speciali.

Nome	Descrizione
`intr`	Invia un segnale di interruzione (‘**SIGINT**’). Normalmente è abbinato al carattere speciale <^*C*>, ovvero alla combinazione di tasti [*Ctrl c*].
`quit`	Invia un segnale di conclusione (‘**SIGQUIT**’). Normalmente è abbinato al carattere speciale <^\>, ovvero alla combinazione di tasti [*Ctrl* \].
`erase`	Cancella l’ultimo carattere digitato.
`kill`	Cancella la riga corrente.
`eof`	Fine del file, ovvero termina l’input. Normalmente è abbinato al carattere speciale <^*D*>, ovvero alla combinazione di tasti [*Ctrl d*].
`stop`	Ferma l’output. Normalmente è abbinato al carattere speciale <^*S*>, ovvero alla combinazione di tasti [*Ctrl s*].
`start`	Riprende l’output dopo uno stop. Normalmente è abbinato al carattere speciale <^*Q*>, ovvero alla combinazione di tasti [*Ctrl q*].
`susp`	Invia un segnale di stop del terminale (‘**SIGTSTP**’), cosa che generalmente fa sì che la shell metta il processo sullo sfondo. Normalmente è abbinato al carattere speciale <^*Z*>, ovvero alla combinazione di tasti [*Ctrl z*].

Per maggiori dettagli sul funzionamento di questo programma, si veda *info stty* oppure *stty(1)*.

14.7.3 Termcap e Terminfo

«

Il primo tipo di terminale, la telescrivente, non poneva problemi particolari di configurazione: la tastiera permetteva di inserire numeri, simboli e caratteri dell'alfabeto inglese, a volte senza poter distinguere tra maiuscole e minuscole, mentre la stampante emetteva un flusso di testo normale, interrotto da un codice di interruzione di riga.

Quando il terminale attuale viene usato ancora in questo modo, non si pongono problemi di configurazione, perché non è importante sapere le dimensioni (in caratteri) dello schermo e non importa sapere come spostare il cursore sullo schermo.

Nel momento in cui si utilizza un programma che sfrutta lo schermo nel modo al quale si è abituati di solito, mostrando bordi, colori, caselline da riempire, si ha la necessità di usare la tastiera anche per spostare il cursore, cancellare, inserire, attivare funzioni speciali. Quindi, lo schermo deve essere in grado di fare di più che visualizzare semplicemente un flusso di caratteri, deve interpretare delle sequenze particolari come la richiesta di utilizzare un colore determinato, di disegnare un bordo, ecc.

Così, la tastiera non serve solo per scrivere lettere, numeri, punteggiatura e terminare le righe con un ritorno a carrello. Adesso occorre utilizzare anche i tasti che spostano il cursore, occorre assegnare funzionalità particolari a tasti che permettono la modifica del testo e a tasti funzionali programmabili.

Riquadro 14.71. Livelli di astrazione nella funzione della tastiera.

Quando si utilizza la tastiera, ciò che si ha di fronte è un dispositivo fisico. La configurazione della tastiera serve a dichiarare in che modo sono abbinati i tasti reali alle funzioni di una tastiera virtuale, con la quale, tra le altre cose, si devono generare delle «combinazioni virtuali» del tipo <*Control_…*>, <*Meta_…*>,...

Dopo avere risolto questo problema, esiste un livello di astrazione successivo: la tastiera virtuale diventa parte del terminale, che, secondo le proprie caratteristiche, può comunicare in modo diverso con i programmi. Da qui nasce la necessità di disporre della configurazione per ogni tipo di terminale da utilizzare.

Inoltre, il terminale in questione, va visto dal punto di vista dell'applicazione finale con cui si intende interagire: se questa viene eseguita localmente, senza intermediazioni di alcun genere, si tratta della console; se invece per comunicare con un'applicazione ci si avvale di un altro programma (per esempio perché si accede da un elaboratore remoto, oppure perché si accede all'elaboratore locale, ma attraverso l'intermediazione di un sistema grafico), occorre vedere in che modo questo programma ulteriore si presenta nei confronti dell'applicazione finale.

Nella storia dell'informatica sono esistiti una quantità enorme di tipi diversi di terminali, intesi come complesso tastiera+schermo, ognuno con piccole differenze rispetto agli altri. Per fare in modo che i programmi che richiedono funzionalità superiori a quelle di una normale telescrivente possano adattarsi ai vari tipi di terminale, viene utilizzato un sistema di configurazione predefinito contenente tutte le informazioni necessarie.

Di questo sistema di configurazione ne esistono due tipi: Termcap e Terminfo. Il primo è il più antico ed è ormai superato, ma viene mantenuto per motivi storici e probabilmente per assicurare la

compatibilità con i programmi più vecchi.

Il sistema Termcap è formato soltanto da un file di testo collocato nella directory '`/usr/share/misc/`' ('`/usr/share/misc/termcap`') e il suo contenuto assomiglia vagamente a quello del file '`/etc/printcap`' (il file di definizione delle stampanti).

Il sistema Terminfo è invece qualcosa di più complesso. È costituito da tanti file, uno per ogni tipo di terminale, distribuiti su varie directory. Il punto di partenza di questa struttura dovrebbe essere la directory '`/usr/share/terminfo/`'.

A partire da '`terminfo/`' si diramano delle directory il cui nome è composto da un solo carattere, corrispondente all'iniziale dei nomi di terminale che contengono. Il listato seguente, mostra solo un estratto minimo di questa struttura.

```
terminfo
|-- 1
|-- 2
|-- 3
…
|-- a
|   `-- ansi
|-- b
|-- c
…
|-- l
|   `-- linux
…
|-- v
|   |-- vt100
|   `-- vt220
…
|-- x
|   `-- xterm
…
```

Se la definizione di un tipo di terminale può essere adatta a diversi nomi, si utilizzano normalmente dei collegamenti simbolici.

I file di definizione del sistema Terminfo sono il risultato di una compilazione attraverso il programma '**tic**',[8] come nell'esempio seguente:

```
# tic prova[Invio]
```

In questo modo, si va a compilare il file '`prova`', generando presumibilmente il file '`/usr/share/terminfo/p/prova`'.

Si ottiene facilmente un elenco dei tipi di terminale previsti con il programma '**toe**':[9]

```
$ toe -a[Invio]

  ansi            ansi/pc-term compatible with color
  dumb            80-column dumb tty
  linux           linux console
  linux+utf8      linux console in utf-8 mode
  rxvt            rxvt terminal emulator (X Window System)
  rxvt-basic      monochrome rxvt terminal emulator (X Window)
  screen          VT 100/ANSI X3.64 virtual terminal
  screen-w        VT 100/ANSI X3.64 virtual terminal with 132
                   cols
  sun             Sun Microsystems Inc. workstation console
  screen-s        VT 100/ANSI X3.64 virtual terminal with
                   hardstatus line
  screen-bce      VT 100/ANSI X3.64 virtual terminal with bce
  vt100           dec vt100 (w/advanced video)
  vt102           dec vt102
  vt220           dec vt220
  vt52            dec vt52
  xterm           X11 terminal emulator
  xterm-debian    Debian xterm (VT220-conformant backspace)
  xterm-xfree86   xterm terminal emulator (XFree86)
  xterm-r5        xterm R5 version
  xterm-r6        xterm X11R6 version
  xterm-vt220     XFree86 xterm emulating vt220
  xterm-color     generic "ANSI" color xterm (X Window System)
  xterm-mono      monochrome xterm
  pcansi          ibm-pc terminal programs claiming to be ansi
  cons25          freebsd console (25-line ansi mode)
  mach            Mach Console
  mach-bold       Mach Console with bold instead of underline
  mach-color      Mach Console with ANSI color
```

La directory ‘`/usr/share/terminfo/`’ è il punto di partenza predefinito per il sistema Terminfo, ma questo può essere alterato uti-

lizzando la variabile di ambiente ***TERMINFO***, per indicare una directory differente. Volendo è possibile personalizzare il sistema Terminfo creando una struttura analoga a partire da '`~/.terminfo/`', cioè dalla directory '`.terminfo/`' nella propria directory personale.

14.7.4 Variabile di ambiente «TERM»

La variabile di ambiente ***TERM*** è il mezzo per definire il tipo di terminale che si utilizza. Normalmente viene impostata automaticamente nel modo più opportuno, con il nome di terminale la cui configurazione deve essere letta da Termcap o da Terminfo.

Quando è impostata in modo errato, si possono presentare due situazioni: il nome del terminale non è previsto, oppure il terminale che si utilizza effettivamente non è compatibile con la definizione contenuta in questa variabile. Nel primo caso, quando si avvia un programma che richiede l'utilizzo di tutto lo schermo, viene segnalato l'errore e, a seconda dei casi, il programma si avvia ugualmente facendo riferimento a un terminale elementare, oppure si rifiuta semplicemente di funzionare.

```
Unknown terminal: pippo
Check the TERM environment variable.
Also make sure that the terminal is defined in the terminfo
database.
```

Se il programma si avvia con una configurazione inappropriata al terminale che si utilizza, questo mostra generalmente un comportamento insolito, per diversi aspetti. Per esempio si possono notare simboli strani sullo schermo, la tastiera potrebbe non rispondere nel modo consueto, lo schermo potrebbe essere ridisegnato solo parzialmente.

14.7.5 Adattabilità di un programma e abilità dell'utilizzatore

«

A questo punto dovrebbe essere chiaro che la tastiera e lo schermo funzionano in maniera differente a seconda di tante condizioni, sia legate alle caratteristiche fisiche, sia relative alle caratteristiche dei programmi attraverso i quali si comunica con le applicazioni finali. Per esempio, il fatto che su una tastiera sia presente il tasto [*Canc*], non vuol dire necessariamente che poi questo dia i risultati che ci si aspetta: la sua pressione potrebbe non avere alcun effetto, oppure generare qualunque altro risultato diverso dal previsto.

Dipende dal nome indicato nel sistema di configurazione dei terminali se questo è in grado di gestire il segnale generato dal tasto [*Canc*] della propria tastiera e se il significato che a questo viene attribuito corrisponde alle aspettative.

Volendo fare un esempio più concreto e anche piuttosto comune, si può provare a confrontare il funzionamento del programma '**mc**' (Midnight Commander), utilizzando la definizione di un terminale differente dal solito, per esempio '**ansi-mono**'.

```
$ TERM=ansi-mono [Invio]

$ export TERM [Invio]
```

Si osserva, prima di tutto, che mancano i colori, che alcune bordature non sono corrette, che i tasti funzionali non danno più l'effetto desiderato.[10]

Alle volte ci si trova veramente davanti a terminali che non possono offrire più di tanto, magari perché si sta operando attraverso una connessione remota con un programma che è in grado di emulare solo

alcuni vecchi tipi di terminale. Allora entrano in gioco due elementi: le alternative offerte dal programma, per cui una stessa cosa può essere ottenuta in modi differenti, per poter essere utilizzato anche in presenza di terminali con poche potenzialità; l'abilità dell'utente di adattarsi alle diverse situazioni.

L'esempio tipico di questo genere di programmi è dato dalle interpretazioni recenti di VI. Quasi tutti questi programmi sono in grado di gestire i tasti freccia, [*Ins*] e [*Canc*]. Ma quando questi non sono disponibili, si può ritornare all'uso tradizionale con i comandi '**h**', '**j**', '**k**' e '**l**', per spostare il cursore, '**i**' e '**x**' per iniziare l'inserimento e per cancellare.

Ciò significa che, quando si studia un nuovo programma, non si devono disdegnare i comandi apparentemente antiquati, perché sono quelli che poi permettono di «tirarsi fuori dai guai».

14.7.6 Ripulitura dello schermo

Esistono due situazioni in cui si può avere la necessità di ripulire lo schermo: quando si scrive uno script con cui si vuole ripulire tutto per mostrare un messaggio all'inizio dello schermo, oppure quando lo schermo sembra impazzito.

Per questo si utilizzano due programmi: '**clear**' e '**reset**'. Questi, in realtà, si avvalgono di un terzo che ha funzioni più generali: '**tput**'. [11]

```
clear
```

Il programma '**clear**' chiama '**tput**' con l'argomento '**clear**', allo scopo di ripulire lo schermo e ricominciare dalla prima posizione in

alto dello schermo.

```
reset
```

Il programma '**reset**' chiama '**tput**' con una serie di argomenti volti a reinizializzare il terminale. È particolarmente utile l'uso di questo programma quando sullo schermo non appaiono più delle lettere normali. In tal caso, si può scrivere '**reset**' e premere [*Invio*] alla cieca. Di solito funziona.

Se si vuole sperimentare questa situazione, basta fare un '**cat**' di un file binario, per esempio un programma qualunque, per non potere più leggere quello che si scrive.

In ogni caso, questi programmi, avvalendosi di '**tput**', funzionano solo in base a quanto conosciuto per mezzo di Terminfo o Termcap. Se la variabile ***TERM*** non contiene il nome corretto, oppure se questo non è presente nel sistema di configurazione dei terminali, a nulla serve un '**reset**'.

Si vedano le pagine di manuale: *tput(1)*, *clear(1)* e *reset(1)*.

Riquadro 14.75. Sequenze di controllo per una console VGA di un sistema GNU/Linux.

Comando	Descrizione
`printf '\033[?2c'`	Definisce un cursore intermittente basso (normale).
`printf '\033[?6c'`	Definisce un cursore intermittente a blocco.

14.7.7 Definizione degli attributi del terminale con «setterm»

Il sistema Terminfo permette di conoscere le stringhe (i comandi) corrispondenti a determinate azioni per il terminale che si utilizza. Attraverso il programma '**setterm**' si può impostare in qualche modo il proprio terminale utilizzando implicitamente tali comandi. La documentazione di '**setterm**', *setterm(1)*, è stringatissima e quindi insufficiente a comprendere bene tutte le possibilità che si avrebbero a disposizione. Tuttavia si tratta di un tipo di intervento sulla gestione del terminale di importanza marginale; quindi non vale la pena di preoccuparsene tanto.

```
setterm opzione
```

Anche se si può utilizzare una sola opzione per volta, quelle disponibili sono molte, ma qui ne vengono descritte solo alcune, tanto da mostrare il senso di questo programma di servizio.

Tabella 14.76. Alcune opzioni.

Opzione	Descrizione
`-repeat [on|off]`	Attiva o disattiva la ripetizione automatica del tasto premuto a lungo. Se non viene specificato l'argomento, si intende attivare l'opzione implicitamente.

Opzione	Descrizione
`-foreground {black|blue|green|cyan|red|magenta|yellow|white|default}`	Permette di modificare il colore di primo piano.
`-background {black|blue|green|cyan|red|magenta|yellow|white|default}`	Permette di modificare il colore dello sfondo.
`-inversescreen [on|off]`	Attiva o disattiva l'inversione dei colori dello schermo. Se non viene specificato l'argomento, si intende attivare l'opzione implicitamente.
`-clear`	Ripulisce lo schermo.
`-reset`	Reinizializza lo schermo.

14.8 Approfondimento: codifica UTF-8

«

La codifica UTF-8 è quella usata universalmente dai sistemi Unix attuali, inclusi i sistemi GNU/Linux; pertanto, è normale attendersi che ogni distribuzione GNU/Linux sia già predisposta correttamente per funzionare in questo modo. Tuttavia, l'attivazione della codifica UTF-8 per una console GNU/Linux comporterebbe qualche compli-

cazione che si vuole affrontare in queste sezioni; inoltre non va trascurato il fatto che rimangono programmi che non si adattano bene a funzionare con una codifica UTF-8.

14.8.1 Caratteri per la console

Per poter visualizzare i simboli dell'insieme di caratteri universale, occorre disporre di questi, indicando al sistema un file che ne contiene le informazioni. Si tratta di file contenuti nella directory '`/usr/share/consolefonts/`' e quelli più adatti per la visualizzazione simultanea degli alfabeti comuni corrispondono al modello '`LatArCyrHeb-`***nn***`.psf`'. In pratica si tratta di un insieme di caratteri che consente la visualizzazione di testi in alfabeto latino, arabo, cirillico ed ebraico. Si carica uno di questi file nel modo seguente:

```
# consolechars ↩
↪-f /usr/share/consolefonts/LatArCyrHeb-16.psf [Invio]
```

Successivamente, la visualizzazione corretta sullo schermo della console richiede anche l'invio di un codice particolare, con l'aiuto del comando '**`printf`**':

```
# printf '\033%G' [Invio]
```

In alternativa, si può indirizzare precisamente al file di dispositivo della console virtuale che deve essere impostata. L'esempio seguente si riferisce a '`/dev/tty1`', ovvero quella che dovrebbe essere la prima console virtuale:

```
# printf '\033%G' > /dev/tty1 [Invio]
```

Si osservi che non sempre le cose funzionano bene. Quando si vogliono caricare insiemi di caratteri del calibro di '`LatArCyrHeb-`

***nn*.psf**', si potrebbe osservare che il testo colorato appare con caratteri errati; per cercare di ovviare a questo inconveniente, si può tentare di caricare prima un insieme comune, quale è quello contenuto nel file '`lat1u-16.psf`'. Praticamente, l'esempio già mostrato andrebbe adattato così:

```
# consolechars -f /usr/share/consolefonts/lat1u-16.psf [Invio]

# consolechars ↩
↪     -f /usr/share/consolefonts/LatArCyrHeb-16.psf [Invio]

# printf '\033%G' [Invio]
```

14.8.2 Mappa della tastiera

Per poter scrivere utilizzando la codifica UTF-8, la mappa della tastiera deve essere stata caricata usando '**loadkeys**' con l'opzione '**-u**', ovvero '**--unicode**'. È possibile adattare la mappa corrente con l'aiuto di '**dumpkeys**', nel modo seguente:

```
# dumpkeys | loadkeys --unicode [Invio]
```

In ogni caso, occorre modificare anche la modalità di funzionamento attraverso '**kbd_mode**':

```
# kbd_mode -u [Invio]
```

Bisogna considerare che le sequenze di composizione non funzionano quando la tastiera è stata configurata in modalità UTF-8. Per la precisione, se sono previste, le sequenze di composizione generano i caratteri, ma lo fanno sempre in byte, pertanto, il loro uso diventa inutile e dannoso, perché generalmente i programmi non sono preparati a gestire sequenze incomplete in UTF-8.

14.8.3 Localizzazione

Naturalmente non bisogna dimenticare di definire la configurazione locale corretta. Prima di poter intervenire nella variabile ***LANG*** ed eventualmente nelle variabili ***LC_**∗***, occorre generare la localizzazione prescelta. Se la propria distribuzione GNU/Linux non fornisce uno strumento più semplice, si può procedere in modo manuale. A partire da '`/usr/share/locale/`' dovrebbero essere disponibili delle sottodirectory che contengono le localizzazioni già definite. Supponendo di voler creare la localizzazione '**`it_IT.UTF-8`**', ci dovrebbe essere la sottodirectory con lo stesso nome: '`it_IT.UTF-8/`'. Se non c'è, la si può creare come nell'esempio seguente:

```
# localedef -v -c -i it_IT -f UTF-8 ↩
↪          /usr/share/locale/it_IT.UTF-8 [Invio]
```

Nel caso di una distribuzione GNU/Linux Debian è più comodo utilizzare il comando seguente:

```
# dpkg-reconfigure locales [Invio]
```

```
.-----------------| Configuring Locales |------------------.
| Locale is a framework to switch between multiple         |
| languages for users who can select to use their language,|
| country, characters, collation order, etc.               |
| Choose which locales to generate.  The selection will be |
| saved to '/etc/locale.gen', which you can also edit      |
| manually (you need to run 'locale-gen' afterwards).      |
| Select locales to be generated.                          |
|    [*] it_CH.UTF-8 UTF-8                                 |
|    [ ] it_IT@euro ISO-8859-15                            |
|    [ ] it_IT ISO-8859-1                                  |
|    [ ] it_IT.UTF-8@euro UTF-8                            |
|    [*] it_IT.UTF-8 UTF-8                                 |
|             <Ok>                  <Cancel>               |
`----------------------------------------------------------'
```

Una volta creata la localizzazione è sufficiente intervenire nella variabile di ambiente ***LANG***, lasciando vuote le altre variabili ***LC_**∗*:

```
$ LANG=it_IT.UTF-8 [Invio]

$ export LANG [Invio]
```

14.8.4 Attivare e disattivare l'utilizzo della codifica UTF-8

«

Il pacchetto Linux console tools[12] contiene due script che dovrebbero facilitare il passaggio rapido alla codifica UTF-8 e il ripristino della codifica normale:

```
unicode_start
```

```
unicode_stop
```

Teoricamente il primo script, che attiva la funzionalità, prevede la possibilità di indicare l'insieme di caratteri da usare; in pratica, forse è meglio definire prima questa cosa.

Vale la pena di vedere cosa fanno questi script. Nei listati seguenti sono state semplificate molte cose:

```
#!/bin/sh
# start unicode mode
dumpkeys | loadkeys --unicode > /dev/null
kbd_mode -u
printf '\033%G'
```

```
#!/bin/sh
# stop unicode
kbd_mode -a
printf '\033%@'
```

Questi script potrebbero essere «rinforzati» aggiungendo la selezione dell'insieme dei caratteri per lo schermo e leggendo la configurazione della mappa della tastiera da un file stabilito. Seguono altri due esempi, sempre molto semplificati, dove in particolare si cerca di inviare il codice di attivazione dello schermo alle prime sei console virtuali:

```
#!/bin/sh
# start console UTF-8
#
kbd_mode -u
loadkeys -c -u /etc/console/boottime.kmap.gz    > /dev/null
for n in 1 2 3 4 5 6
do
    if [ -w /dev/tty$n ]
    then
        printf '\033%G' > /dev/tty$n
    fi
done
/usr/bin/consolechars -f \
    /usr/share/consolefonts/lat1u-16.psf.gz
/usr/bin/consolechars -f \
    /usr/share/consolefonts/LatArCyrHeb-16.psf.gz
```

```
#!/bin/sh
# stop console UTF-8
#
kbd_mode -a
loadkeys /etc/console/boottime.kmap.gz  > /dev/null
for n in 1 2 3 4 5 6
do
    if [ -w /dev/tty$n ]
    then
        printf '\033%@' > /dev/tty$n
    fi
done
/usr/bin/consolechars -f \
    /usr/share/consolefonts/lat1u-16.psf.gz
```

Questi comandi possono essere utilizzati dall'utente '**root**' o da un utente comune, ma nel secondo caso, l'effetto del cambiamento di funzionamento si trasmette in modo incompleto alle console, perché

possono mancare i permessi di scrittura ai file di dispositivo delle altre console virtuali. Si osservi che, una volta deciso di configurare la console per l'uso della codifica UTF-8, sarebbe meglio evitare di riportarla a funzionare secondo il vecchio sistema a byte, perché comunque c'è la possibilità di avviare i programmi più vecchi con l'aiuto di '**luit**'.

Riquadro 14.82. Configurazione automatica della console per la codifica UTF-8.

Considerato che UTF-8 consente la rappresentazione di tutti simboli previsti dall'insieme di caratteri universale, è auspicabile che diventi presto lo standard di tutti. Pertanto, è evidente che i passaggi necessari a impostare il terminale della console di un sistema GNU/Linux vadano inseriti all'interno di uno script della procedura di inizializzazione del sistema, in modo che all'avvio sia tutto eseguito automaticamente. Tuttavia, non è detto che le cose funzionino come previsto. Infatti, se si usa un sistema come uDev per la generazione automatica dei file di dispositivo nella directory '`/dev/`', può darsi che nella fase di avvio non ci siano tutti i file riferiti alle console virtuali che si intendono attivare. Per fare in modo che le cose funzionino, anche di fronte a questa difficoltà, può essere il caso di mettere i comandi necessari nel file '`/etc/profile`', in modo che siano eseguiti ogni volta che un utente accede al sistema, anche se questo è controproducente quando si vuole lavorare in una console secondo la vecchia modalità a byte.

14.8.5 Utilizzo di «luit»

«

Il programma '**luit**', il quale fa parte di X, 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 stata predisposta.

```
$ 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 prende il sopravvento su tutte le variabili ***LC_**** e su ***LANG***.

14.9 Console grafica VGA nei sistemi GNU/Linux

La console di un sistema GNU/Linux è normalmente priva di funzioni grafiche, con una risoluzione di 25 righe per 80 colonne di caratteri. Tuttavia, considerate le dimensioni comuni di uno schermo attuale, diventa auspicabile la possibilità di visualizzare una densità maggiore di caratteri.

In generale, si ritiene consigliabile e necessario solo l'utilizzo delle funzioni di programmazione del tipo di carattere per lo schermo, in modo da poter gestire simboli di alfabeti non latini; tutto il resto, anche la gestione del *frame buffer*, sarebbe meglio riservarlo alle situazioni in cui dovesse rivelarsi indispensabile.

14.9.1 Parametri di avvio: «vga=ask»

All'avvio di un kernel Linux in un elaboratore provvisto di adattatore grafico VGA, è possibile utilizzare il parametro '**vga**', per richiedere la visualizzazione sullo schermo della console a una densità maggiore di caratteri:

```
vga=n
```

```
vga=ask
```

Il valore numerico che si può assegnare al parametro '**vga**' dipende dalle caratteristiche dell'adattatore grafico. Pertanto, generalmente si usa assegnare la parola chiave '**ask**', con la quale si ottiene un menù dal quale è possibile selezionare ciò che è disponibile effettivamente.

L'utilizzo con successo di questo parametro di avvio implica che durante il funzionamento del sistema operativo non si modifichino i caratteri predefiniti (si veda la sezione successiva), cosa che però è quasi necessario fare, per poter gestire qualche simbolo in più, oltre a quelli essenziali delle lingue occidentali.

14.9.2 Modifica dei caratteri

«

Le console virtuali, che normalmente utilizzano schermi VGA, possono essere configurate in modo da utilizzare un insieme di caratteri differente da quello standard (il famigerato CP437) e anche per permettere la visualizzazione di più righe e più colonne.

In passato, nei sistemi GNU/Linux è stato usato il programma '**setfont**',[13] ora sostituito con '**consolechars**', [14] per l'impostazione dei caratteri da mostrare sullo schermo di una console:

```
setfont [opzioni] file_di_configurazione
```

```
consolechars [opzioni]
```

È molto importante l'uso di '**setfont**' o di '**consolechars**' quando si decide di utilizzare un insieme di caratteri esteso, come ISO 8859-1 o meglio ancora come ISO 10646, per poter visualizzare caratteri come le lettere accentate maiuscole, le quali non fanno parte della codifica standard di un'interfaccia video a caratteri tipica.

Per ottenere il risultato, questi programmi si avvalgono di file di definizione dei caratteri, collocati nella directory '`/usr/share/consolefonts/`'.

L'esempio seguente, visto sia per '**setfont**', sia per '**consolechars**', serve a ottenere la visualizzazione di caratteri utili nelle lingue europee, incluso l'arabo e l'ebraico, in uno schermo composto da 25 righe.

```
# setfont /usr/share/consolefonts/LatArCyrHeb-16.psf [Invio]

# consolechars ↩
↪-f /usr/share/consolefonts/LatArCyrHeb-16.psf [Invio]
```

Eventualmente, se la dimensione dei caratteri non è quella desiderata, si possono provare altri file della famiglia '`LatArCyrHeb-*.psf`'.

Per approfondire la sintassi di questi programmi, si veda la pagina di manuale *setfont(8)* oppure *consolechars(8)*; inoltre si veda la sezione 14.8.1.

14.9.3 Console VGA e «frame buffer»

«

Il kernel Linux è in grado di gestire graficamente la console, attraverso ciò che viene chiamato *frame buffer*. In pratica, nel kernel vengono incluse delle funzionalità apposite, assieme al codice specifico per l'adattatore grafico da controllare, di solito in forma di moduli (si veda eventualmente la sezione 8.3.8.10).

Se non si vuole avere questa gestione già attiva in fase di avvio, è consigliabile che queste funzioni risiedano all'interno di moduli del kernel, in modo da poter attivare esattamente ciò che serve al proprio adattatore grafico. Prima di farlo, però, occorre accertarsi di disporre del file di dispositivo '`/dev/fb0`'. Eventualmente, lo si può creare così:

```
# mknod -m 660 /dev/fb0 c 29 0 [Invio]
```

Il primo modulo da attivare per mettere in funzione il *frame buffer* è '**fbcon**':

```
# modprobe fbcon [Invio]

# lsmod | grep fbcon [Invio]

fbcon                  37024  0
tileblit                2688  1 fbcon
font                    8320  1 fbcon
bitblit                 5376  1 fbcon
```

Dopo il caricamento di questo modulo, non si dovrebbe osservare alcun cambiamento per quanto riguarda l'aspetto dello schermo. La fase successiva richiede il caricamento del modulo specifico per l'adattatore grafico, che prima va identificato:

```
# lspci | grep VGA [Invio]
```

```
0000:01:00.0 VGA compatible controller: nVidia Corporation ↩
↪NV5M64 [RIVA TNT2 Model 64/Model 64 Pro] (rev 15)
```

Quindi si cerca il modulo nella directory '`/lib/modules/`***versione***`/kernel/drivers/video/`':

```
# uname -a[Invio]
```

```
Linux nanohost 2.6.15.6 #1 PREEMPT Mon Mar 6 16:05:13 CET ↩
↪2006 i686 GNU/Linux
```

```
# cd /lib/modules/2.6.15.6/kernel/drivers/video[Invio]
```

```
# find . -name \*fb\* -print[Invio]
```

```
./cyber2000fb.ko
./arcfb.ko
./aty/aty128fb.ko
./aty/atyfb.ko
…
./neofb.ko
./nvidia/nvidiafb.ko
./pm2fb.ko
…
./vga16fb.ko
./vfb.ko
```

Intuitivamente, si opta per il modulo '**`nvidiafb`**':

```
# modprobe nvidiafb[Invio]
```

```
# lsmod | grep fb[Invio]
```

```
nvidiafb               48284  1
fbcon                  37024  73
tileblit                2688  1 fbcon
font                    8320  1 fbcon
bitblit                 5376  1 fbcon
```

Se si dispone del programma '**fbset**', si può tentare di modificare al volo la risoluzione. Per farlo, occorre leggere prima il contenuto del file '/etc/fb.modes':

```
# grep ^mode /etc/fb.modes [Invio]

mode "640x480-60"
mode "640x480-72"
mode "640x480-75"
mode "640x480-90"
mode "640x480-100"
mode "768x576-75"
mode "800x600-48-lace"
mode "800x600-56"
mode "800x600-60"
mode "800x600-70"
mode "800x600-72"
mode "800x600-75"
mode "800x600-90"
mode "800x600-100"
mode "1024x768-43-lace"
mode "1024x768-60"
mode "1024x768-70"
mode "1024x768-72"
mode "1024x768-75"
mode "1024x768-90"
mode "1024x768-100"
mode "1152x864-43-lace"
mode "1152x864-47-lace"
```

```
mode "1152x864-60"
mode "1152x864-70"
mode "1152x864-75"
mode "1152x864-80"
mode "1280x960-75-8"
mode "1280x960-75"
mode "1280x960-75-32"
mode "1280x1024-43-lace"
mode "1280x1024-47-lace"
mode "1280x1024-60"
mode "1280x1024-70"
mode "1280x1024-74"
mode "1280x1024-75"
mode "1600x1200-60"
mode "1600x1200-66"
mode "1600x1200-76"
```

Si tenta di utilizzare una risoluzione di '**800x600-60**':

```
# fbset 800x600-60 [Invio]
```

In alternativa, si può indicare la risoluzione e la quantità di colori, senza specificare la frequenza di scansione. In questo caso per i colori e le sfumature si usano 16 bit:

```
# fbset 800 600 800 600 16 [Invio]
```

L'utilizzo del *frame buffer* ha degli inconvenienti: pur usando i moduli, una volta caricati, non è più possibile ritornare alla gestione normale; se si commettono degli errori, si rischia di rendere invisibile e inutilizzabile la console.

14.9.4 «Frame buffer» dall'avvio

«

Nella sezione precedente è stato mostrato, a grandi linee, il modo in cui è possibile attivare la gestione della console grafica con il *frame buffer*, quando il sistema è già in funzione. Per fare in modo che questa modalità di funzionamento della console sia attiva già al momento dell'avvio del sistema, occorre utilizzare delle opzioni di avvio per il kernel Linux.

Quando si sceglie la strada delle opzioni di avvio, occorre conoscere perfettamente le caratteristiche del proprio adattatore grafico, ma in generale, a meno di avere un adattatore troppo sofisticato o troppo vecchio, conviene usare la modalità standard VESA 2.0.

La gestione della console grafica deve essere già disponibile nel corpo principale del kernel, sia per quanto riguarda la gestione generica della console in modo grafico (che si potrebbe mettere nel modulo '**fbcon**'), sia per la gestione dello standard VESA (il modulo sarebbe '**vesafb**'). Quindi, per queste due cose non si possono usare i moduli.

Quando il kernel è pronto, è sufficiente l'opzione di avvio '**vga=*n***', dove ***n*** va sostituito con un numero, come descritto nella tabella successiva.

Tabella 14.89. Modalità VESA da usare con l'opzione di avvio '**vga=*n***', quando la gestione grafica della console in modalità VESA è inclusa nel kernel.

Profondità di colori	320×200	640×400	640×480	800×500	800×600	896×672
4 bit (16 colori)					770	
8 bit (256 colori)		768	769	879	771	815

Profondità di colori	320×200	640×400	640×480	800×500	800×600	896×672
15 bit	781	801	784	880	787	816
16 bit	782	802	785	881	788	817
24 bit	783	803	786	882	789	818
32 bit		804	809	883	814	819

Profondità di colori	1024 ×640	1024 ×768	1152 ×720	1280 ×1024	1440 ×900	1600 ×1200
4 bit (16 colori)		772		774		
8 bit (256 colori)	874	773	869	775	864	796
15 bit	875	790	870	793	865	797
16 bit	876	791	871	794	866	798
24 bit	877	792	872	795	867	799
32 bit	878	824	873	829	868	834

Nelle situazioni più comuni si utilizza l'opzione '**`vga=791`**' per una risoluzione a 1024×768, ottenendo un carattere abbastanza visibile con gli schermi usuali.

Se il proprio adattatore grafico non è compatibile a sufficienza con lo standard VESA 2.0, non è possibile avvalersi di tale funzionalità nel modo descritto, ed eventualmente si può inibire un'opzione già definita con l'opzione '**`vga=normal`**'. Se si vuole usare la console grafica in ogni caso, partendo dall'avvio, occorre usare opzioni adeguate per il proprio adattatore grafico, il cui codice deve essere incorporato nella porzione principale del kernel Linux.

Per le informazioni dettagliate sulle opzioni di avvio relative ai vari adattatori grafici, va consultata la documentazione del kernel Linux: '***sorgenti_linux***`/Documentation/fb/*`'.

14.10 Utilizzo del dispositivo di puntamento

«

Il mouse, in un terminale a caratteri, non è una cosa tanto comune. È normale in un ambiente grafico, ma nel caso di GNU/Linux c'è la possibilità di usarlo anche nelle console virtuali. Per gestire un mouse in questa situazione è necessario un demone che si occupi di seguirlo e di fornire ai programmi le informazioni sulle azioni del mouse stesso. Si tratta in pratica di un servente per la gestione del mouse. Trattandosi di un servente, i programmi con cui si può interagire con il mouse sono dei clienti e dipendono dal servente per il tipo di comunicazione che tra loro deve instaurarsi.

Il servente utilizzato normalmente per GNU/Linux è il demone '**gpm**', il quale ha in particolare il vantaggio di essere utile anche con i programmi che non sono fatti per il mouse, per copiare e incollare del testo.

In alcune situazioni, la gestione del mouse può diventare conflittuale, per esempio quando si utilizza un cosiddetto mouse bus (*bus-mouse*). In questa situazione non è possibile avere più programmi che leggono contemporaneamente il dispositivo corrispondente al mouse; in pratica non ci può essere in funzione il demone '**gpm**' assieme al sistema grafico X (a meno che X si avvalga proprio di '**gpm**') e nemmeno possono essere messi in funzione più sistemi grafici contemporaneamente. Il demone '**gpm**' è in grado di risolvere il problema occupandosi da solo del mouse e passando a tutte le altre applicazioni eventuali le informazioni sulle azioni compiute con il mouse stesso.

14.10.1 Dispositivo del mouse

Per convenzione, il file ‘`/dev/mouse`’ dovrebbe corrispondere al dispositivo del mouse. In pratica, si potrebbe creare un collegamento simbolico con questo nome che punta al dispositivo corrispondente al mouse utilizzato effettivamente.

Nel caso particolare dei mouse seriali, cioè di quelli connessi a una porta seriale, sono stati usati in passato i dispositivi ‘`/dev/cua*`’. Attualmente, questi sono superati e al loro posto si fa riferimento ai corrispondenti ‘`/dev/ttyS*`’.

Quando la lettura di questo dispositivo può essere solo esclusiva, a causa della sua natura, per evitare conflitti tra i programmi nel modo descritto in precedenza, si può creare il file FIFO ‘`/dev/gpmdata`’. Questo viene gestito dal demone ‘**gpm**’ allo scopo di fornire a tutti gli altri programmi che accedono direttamente al mouse le informazioni sulle azioni compiute con lo stesso.

```
# mknod /dev/gpmdata p[Invio]
```

Il comando appena mostrato è ciò che serve per creare questo file nel caso non sia già disponibile. Per fare in modo che ‘**gpm**’ gestisca questo file e di conseguenza si occupi del mouse in qualunque situazione, deve essere utilizzata l’opzione ‘**-R**’. Inoltre, se si utilizza il sistema grafico X è necessario modificare manualmente la sua configurazione (‘`/etc/X11/xorg.conf`’) nella sezione ‘**InputDevice**’, più o meno come si vede nell’esempio seguente:

```
Section "InputDevice"
    Identifier  "Generic Mouse"
    Driver      "mouse"
    Option      "CorePointer"
    Option      "Device"                "/dev/gpmdata"
    Option      "Protocol"              "IntelliMouse"
    Option      "Emulate3Buttons"       "true"
    Option      "ZAxisMapping"          "4 5"
    Option      "Buttons"               "5"
EndSection
```

In pratica, per il sistema grafico X e per qualunque altro programma che dovesse accedere al dispositivo del mouse direttamente, si deve fare riferimento al tipo di mouse '**`IntelliMouse`**', utilizzando il file di dispositivo '`/dev/gpmdata`'.

Tabella 14.91. Alcuni file riferiti a dispositivi di puntamento secondo il kernel Linux. L'elenco completo può essere consultato nel file '***sorgenti_linux***`/Documentation/devices.txt`' tra i sorgenti del kernel.

File di dispositivo	Descrizione
'`/dev/mouse`'	Collegamento simbolico al file di dispositivo oppure a un file FIFO adatto.
'`/dev/gpmdata`'	File FIFO standard per ritrasmettere il movimento del mouse a più programmi.
'`/dev/logibm`'	mouse bus Logitech.
'`/dev/psaux`'	mouse PS/2, o mouse USB compatibile.
'`/dev/usb/mouse`***n***'	***n***-esimo mouse USB (a partire da zero), non compatibile con il tipo PS/2.

14.10.2 Utilizzo di «gpm»

Il programma '**gpm**'[15] permette di copiare e incollare porzioni dello schermo con i programmi normali e fornisce a quelli predisposti l'accesso a tutte le funzionalità del mouse. Può essere messa in funzione una sola copia del programma alla volta, di conseguenza è normale che '**gpm**' venga avviato una volta per tutte attraverso la procedura di inizializzazione del sistema.

```
gpm [opzioni]
```

A meno di fare uso di opzioni particolari, '**gpm**' si aspetta di trovare il collegamento '`/dev/mouse`' che punti al file di dispositivo corrispondente al mouse effettivamente a disposizione.

Se '**gpm**' viene utilizzato con l'opzione '**-R**', allora si abilita la gestione del file FIFO '`/dev/gpmdata`' e tutti gli altri programmi che dovessero accedere direttamente al mouse dovrebbero utilizzare questo file come dispositivo (che, salvo altra indicazione, si comporta come quello di un mouse '**MouseSystems**').

Tabella 14.92. Alcune opzioni.

Opzione	Descrizione
`-B` ***sequenza***	Con questa opzione è possibile definire la disposizione dei tasti. Per esempio, '**`gpm -B 123`**' indica di utilizzare i tasti nella posizione normale: il primo è quello a sinistra, il secondo è quello centrale e il terzo è quello a destra. Nello stesso modo si può indicare una disposizione inversa per facilitare un utente che preferisce usare la mano sinistra ('**`gpm -B 321`**').
`-m` ***file***	Permette di indicare un file di dispositivo diverso dal solito '`/dev/mouse`'.
`-R` [***tipo***]	Abilita la gestione del file FIFO '`/dev/gpmdata`' allo scopo di fornire ad altre applicazioni, che accedono direttamente al mouse, le informazioni sulle sue azioni. Se si indica il tipo, questo specifica il protocollo di comunicazione da utilizzare per tale scopo; altrimenti si fa riferimento in modo predefinito al tipo '**`MouseSystems`**' ('**`msc`**'). In base agli esempi mostrati, si dovrebbe invece specificare il tipo '**`IntelliMouse`**' ('**`ms3`**').
`-t` ***tipo***	Permette di indicare il tipo di mouse a disposizione. Quando non si specifica questa opzione, il tipo predefinito è '**`ms`**', corrispondente a un mouse Microsoft con due o tre tasti. In particolare, '**`-t help`**' elenca tutti i tipi disponibili (si veda la tabella 14.93).
`-2`	Forza un funzionamento a due tasti. In questo modo il primo tasto serve a evidenziare e l'altro a incollare.

Opzione	Descrizione
`-3`	Forza un funzionamento a tre tasti. In questo modo il primo tasto serve a evidenziare, il secondo a incollare e il terzo a estendere la zona evidenziata. Questo è il funzionamento predefinito, perché il secondo tasto viene attivato solo a partire dal momento in cui questo viene premuto. Perciò, normalmente, non occorre preoccuparsi di indicare quanti tasti utilizzare.
`-S` ***comandi***	Permette di definire dei comandi da eseguire in corrispondenza di un clic triplo sul primo e sul terzo tasto.

Tabella 14.93. Elenco di alcuni nomi dei tipi di mouse utilizzabili con l'opzione '**`-t`**' e con l'opzione '**`-R`**'.

Tipo	Sinonimo	Descrizione
`mman`	`Mouseman`	Mouseman.
`ms`		Microsoft a due o tre tasti e compatibili (predefinito).
`ms+`		Come '**`ms`**', con il trascinamento nel tasto centrale.
`bare`	`Microsoft`	Microsoft a due tasti.
`msc`	`MouseSystems`	Mouse System, tre tasti.
`sun`		Variante del Mouse System.
`mm`	`MMSeries`	
`logi`	`Logitech`	Alcuni mouse seriali Logitech.

Tipo	Sinonimo	Descrizione
`logim`		Mouse Logitech che devono funzionare come mouse Mouse System a tre tasti.
`bm`	`BusMouse`	Busmouse Microsoft e compatibili.
`ps2`	`PS/2`	Busmouse PS/2.
`ncr`		Alcune penne di puntamento di alcuni portatili (NCR3125pen).
`wacom`		Tavoletta Wacom.
`genitizer`		Tavoletta Genitizer.
`logim`		Mouse Logitech in cui abilitare il funzionamento in modalità Mouse System.
`pnp`		Microsoft pnp.
`imps2`		Microsoft IntelliMouse su porta PS/2.
`ms3`		Mouse seriali IntelliMouse a tre tasti.
`netmouse`		Genius NetMouse (due tasti normali, più un tasto «su» e un tasto «giù»).
`cal`		Calcomp Ultraslate.
`calr`		Calcomp Ultraslate in modalità relativa.
`twid`		Tastiera Twidder.

Tipo	Sinonimo	Descrizione
`syn`	`synaptics`	Touchpad seriale Synaptics.
`syn2`	`synaptics_ps2`	Touchpad PS/2 Synaptics.
`brw`		Mouse Fellowes Browser a quattro bottoni e una rotella.
`js`	`Joystick`	Emulazione del mouse attraverso un joystick.
`summa`		Tavoletta Summa/Genius.
`mtouch`		Schermi *touchscreen* MicroTouch.
`acecad`		Tavolette Acecad in modalità assoluta.
`kmiabps2`		Kensignton Mouse «in a box» su PS/2.

Il funzionamento di '**`gpm`**' è relativamente semplice. Quando il mouse è riconosciuto dal programma che si sta utilizzando, dipende da questo il modo di gestire e interpretare le azioni compiute con il mouse. Quando il programma non è in grado di controllare il mouse, è comunque possibile utilizzare la funzione di copia del testo.

Si seleziona una zona dello schermo premendo il primo tasto e trascinando fino alla posizione finale. Per incollare si può cambiare console virtuale, raggiungendo così l'applicazione all'interno della quale incollare il testo, quindi si preme il secondo tasto, o in mancanza il terzo. Il testo viene inserito come se fosse digitato, di conseguenza occorre che il programma lo permetta.

L'opzione '**`-S`**' permette di definire tre comandi, separati con il sim-

bolo due punti (‘**:**’), da eseguire in occasione di un clic triplo con il primo e il terzo tasto. In pratica, si tiene premuto il primo o il terzo tasto, mentre con l’altro (il terzo o il primo rispettivamente) si esegue un clic triplo in rapida successione. Se entro tre secondi dal rilascio dei tasti viene premuto uno dei tre tasti, viene eseguito uno dei comandi indicati nell’argomento di questa opzione.

Per esempio, se si utilizza l’opzione ‘**-S "echo ciao:echo hello:e**
e si preme un clic triplo, del tipo descritto, seguito dalla pressione del primo tasto, si ottiene l’esecuzione di ‘**echo ciao**’, cioè viene visualizzata la parola ‘**ciao**’. Se invece alla fine si seleziona il secondo tasto, si ottiene la parola ‘**hello**’. Infine, se si tratta del terzo tasto, si ottiene ‘**bye**’.

Questo sistema potrebbe essere particolarmente utile per definire un comando per il riavvio del sistema, quando per qualche motivo non si può usare la tastiera per farlo e non si rendono disponibili altre alternative.

Segue la descrizione di alcuni esempi.

- # **gpm -t imps2** [*Invio*]

  Avvia ‘**gpm**’ predisponendolo per utilizzare un mouse PS/2 a tre tasti con rotellina (va bene anche se la rotellina non c’è e se i tasti sono solo due).

- # **gpm -R -t imps2** [*Invio*]

  Avvia ‘**gpm**’ predisponendolo per utilizzare un mouse PS/2 a tre tasti con rotellina, abilitando la gestione del file ‘`/dev/gpmdata`’. Il sistema grafico X e altri programmi che dovessero accedere direttamente al dispositivo del mouse, dovrebbero esse-

re istruiti a utilizzare il file ‘`/dev/gpmdata`’, corrispondente a un mouse ‘**IntelliMouse**’.

- `# gpm -t imps2 -m /dev/psaux -R ms3` [*Invio*]

  Come nell’esempio precedente, avvia ‘**gpm**’ predisponendolo per utilizzare un mouse PS/2 a tre tasti con rotellina, abilitando la gestione del file ‘`/dev/gpmdata`’; in particolare viene specificato il file di dispositivo del mouse e il tipo di protocollo da usare per la comunicazione attraverso il file ‘`/dev/gpmdata`’.

- `# gpm -S "shutdown -h now:shutdown -r now:init 0"` [*Invio*]

  Avvia ‘**gpm**’ definendo i comandi speciali da eseguire in caso di un clic triplo. Se dopo il clic triplo si preme il primo tasto, si conclude l’attività del sistema; se si preme il secondo, si riavvia; se si preme il terzo, si conclude l’attività, ma attraverso una chiamata diretta all’eseguibile ‘**init**’.

14.10.3 Avvio del servizio di gestione del mouse

Si è accennato al fatto che il demone ‘**gpm**’ venga avviato normalmente dalla procedura di inizializzazione del sistema, nel modo già stabilito dalla stessa distribuzione GNU/Linux che si utilizza. Se si vogliono gestire funzionalità speciali di ‘**gpm**’, come per esempio il file FIFO ‘`/dev/gpmdata`’, cosa che si ottiene con l’opzione ‘**-R**’, occorre intervenire nello script che avvia questo demone.

Alcune distribuzioni, prevedono un file di configurazione contenente l’assegnamento di variabili di ambiente che poi vengono incorporate e utilizzate nello script di avvio del servizio ‘**gpm**’. Tuttavia potrebbe non essere stata prevista la possibilità di aggiungere delle opzioni ulteriori; in tal caso si deve intervenire direttamente nello script.

In particolare, la distribuzione Red Hat gestisce il servizio attraverso lo script '`/etc/rc.d/init.d/gpm`', mentre la distribuzione Debian usa il file '`/etc/init.d/gpm`'. Inoltre, la distribuzione Debian mette a disposizione lo script '**`gpmconfig`**' per facilitare l'intervento nel file di configurazione, corrispondente a '`/etc/gpm.conf`'.

14.11 Monitoraggio di una sessione di lavoro

«

L'attività svolta durante una sessione di lavoro attraverso un terminale potrebbe essere registrata volontariamente in modo da annotare le operazioni svolte, eventualmente anche a titolo di prova, come potrebbe essere l'esecuzione di un test di esame.

In aggiunta, le console virtuali di GNU/Linux possono essere osservate attraverso dei dispositivi appositi: '`/dev/vcs*`'.

14.11.1 Utilizzo di «script»

«

Il programma '**`script`**'[16] permette di registrare la sessione di lavoro svolta attraverso un terminale a caratteri. Si avvia il programma e questo, a sua volta, avvia una copia della shell predefinita; da quel momento, tutto ciò che viene digitato ed emesso attraverso il terminale viene memorizzato in un file. Il file può essere indicato nella riga di comando, altrimenti viene creato il file '`typescript`' nella directory corrente.

```
script [-a] file
```

L'opzione '**`-a`**' permette di continuare la registrazione in un file già utilizzato in precedenza, senza cancellarlo inizialmente.

Per terminare l'esecuzione della registrazione della sessione di lavoro, basta concludere l'attività della shell avviata da '**`script`**'; di solito si tratta di utilizzare il comando '**`exit`**'.

14.11.2 File di dispositivo «/dev/vcs*»

I file di dispositivo '`/dev/vcs*`', definiti *Virtual console capture device*, possono essere usati per visualizzare lo schermo di una console particolare. Il meccanismo è estremamente banale, in quanto basta leggere il loro contenuto: in ogni momento, il risultato che si ottiene da questa lettura è l'immagine dello schermo di quella console particolare che quel dispositivo rappresenta.

```
# cat /dev/vcs1 [Invio]
```

L'esempio mostra la visualizzazione del contenuto dello schermo della prima console virtuale, corrispondente al dispositivo '`/dev/tty1`', dell'istante in cui si esegue il comando.

In particolare, il dispositivo '`/dev/vcs0`' fa riferimento alla console virtuale attiva, mentre i file contrassegnati da un numero finale (diverso da zero) corrispondono alle rispettive console virtuali, identificate in modo preciso tramite quel numero.

14.12 Strumenti per la gestione delle console virtuali

Le console virtuali di GNU/Linux sono gestite normalmente attraverso la configurazione del file '`/etc/inittab`', in cui, a seconda del livello di esecuzione, si attivano diversi programmi Getty abbinati ad altrettanti terminali o console virtuali. Generalmente, in que-

sto modo, non vengono utilizzate tutte le console virtuali possibili, pertanto quelle rimanenti potrebbero essere sfruttate per altri scopi.

Le console virtuali disponibili possono essere utilizzate per visualizzare in modo continuo informazioni utili sul funzionamento del sistema, come per esempio quelle provenienti da un file per le registrazioni del sistema (*log*).

```
# tail -f /var/log/messages > /dev/tty10 & [Invio]
```

L'esempio mostra l'utilizzo di '**tail**' per visualizzare la fine del file '/var/log/messages' e tutte le righe che gli vengono aggiunte successivamente. Invece di impegnare il terminale dal quale viene avviato, il comando viene messo sullo sfondo ('**&**') e l'output viene emesso attraverso la decima console virtuale (che si presume sia disponibile).

14.12.1 Utilizzo di «open»

«

Il programma '**open**'[17] permette di avviare un comando in una nuova console virtuale (non utilizzata precedentemente). Per distinguere il comando dalle opzioni di '**open**' si utilizza un trattino doppio ('**--**') per segnalare l'inizio del comando stesso.

```
open [opzioni] [--] comando [opzioni_del_comando]
```

Tabella 14.94. Alcune opzioni.

Opzione	Descrizione
`-c` ***n***	Questa opzione permette di definire esplicitamente quale console virtuale utilizzare attraverso l'argomento che indica il numero di questa (le console virtuali sono numerate a partire da uno).
`-l`	Fa in modo che il comando venga trattato come se fosse una «shell di *login*», cioè una shell avviata dalla procedura di accesso (dopo che l'autenticazione dell'utente è avvenuta con successo). Questo comporta l'aggiunta di un trattino ('**-**') davanti al nome del comando.
`--`	Segna la fine delle opzioni di '**open**' e l'inizio del comando. È necessario l'uso di questo trattino doppio quando il comando da eseguire ha, a sua volta, degli argomenti.

Segue la descrizione di alcuni esempi.

- `#` **`open bash`** [*Invio*]

  Avvia l'eseguibile '**bash**' nella prima console virtuale libera.

- `#` **`open -l bash`** [*Invio*]

  Avvia l'eseguibile '**bash**' nella prima console virtuale libera, trattando il processo relativo come una shell di *login*.

- `#` **`open -c 10 -l bash`** [*Invio*]

  Come nell'esempio precedente, utilizzando espressamente la decima console virtuale.

- `#` **`open -- ls -l`** [*Invio*]

  Esegue il comando '**ls -l**' utilizzando la prima console virtuale libera. In questo caso, dovendo indicare un comando con argo-

menti, è stato inserito il trattino doppio per segnalare l'inizio del comando stesso.

14.12.2 Utilizzo di «switchto»

Il programma '**switchto**' [18] permette di selezionare una console virtuale particolare. Può essere utile in uno script.

```
switchto n
```

L'esempio seguente mostra il passaggio all'undicesima console virtuale:

```
# switchto 11 [Invio]
```

14.13 Terminali virtuali, o finestre, con il programma Screen

È già stato descritto più volte il funzionamento delle console virtuali di GNU/Linux, che, attraverso una sola console fisica, permettono la gestione di più sessioni di lavoro differenti, a cui si accede generalmente con le combinazioni di tasti [*Ctrl Fn*], oppure [*Ctrl Alt Fn*]. Un effetto simile si può ottenere attraverso dei programmi, utilizzabili anche quando non si dispone di una console GNU/Linux.

Un programma che svolga questo compito non è così comodo da utilizzare come può esserlo una console virtuale, però può offrire delle possibilità in più. Per esempio, potrebbe trasferire il ***terminale virtuale*** su un altro terminale fisico, senza dover sospendere, né interrompere, il lavoro che si stava svolgendo. In pratica, l'unico programma che si utilizzi per questo scopo è Screen,[19] il quale permette

di fare una quantità di cose, anche il trasferimento di un terminale virtuale a un altro utente (consentendo a questo di continuare il lavoro).

Lo studio di Screen è impegnativo come lo è l'approfondimento di una shell sofisticata. Qui si vogliono mostrare solo i rudimenti, trascurando volutamente funzionalità che, se utilizzate, richiederebbero attenzione per ciò che riguarda la sicurezza.

14.13.1 Funzionamento e organizzazione generale

Screen, costituito precisamente dell'eseguibile '**screen**', è un programma che si interpone tra una shell (o un applicativo diverso) e il terminale utilizzato effettivamente. In pratica, si tratta di un gestore di finestre a caratteri che, tra le altre cose, permette di aprire più sessioni contemporanee utilizzando un solo terminale fisico.

Ogni terminale virtuale, ovvero ogni finestra, mette a disposizione le funzionalità di un terminale VT100 con delle estensioni di vario tipo. Per ogni finestra viene conservato uno storico delle ultime righe visualizzate, permettendo lo scorrimento all'indietro e la copia di porzioni di questo all'interno dello standard input della stessa o di un'altra finestra.

Come si può intuire, per accedere alle funzionalità offerte da Screen occorre utilizzare dei comandi composti da combinazioni di tasti che vengono intercettati da questo, senza essere passati all'applicazione sottostante, provocando così un'alterazione del comportamento normale di queste applicazioni.

Spesso, viene attivato il bit SUID al binario '**screen**', assieme all'attribuzione della proprietà all'utente '**root**'. Ciò permette a

Screen di fare delle cose molto comode, ma richiede attenzione nella sua configurazione, perché ciò potrebbe tradursi in un pericolo in più per chi lo utilizza. Se non si vuole approfondire tanto l'uso di Screen, sarebbe meglio togliere tale permesso.

```
# chmod ug-s /usr/bin/screen [Invio]
```

Se Screen è in condizione di poterlo fare (di solito solo se è attivato il bit SUID per il binario '**`screen`**' e questo appartiene all'utente '**`root`**'), aggiorna il file '`/etc/utmp`', cosa che consente di tenere traccia anche di tutti i terminali virtuali aperti attraverso di esso.

Per poter funzionare, Screen deve creare un file FIFO (*pipe* con nome), per ogni gruppo di finestre aperto, cioè per ogni terminale fisico a cui è connesso effettivamente. Tale file viene definito «socket» da Screen e dalla sua documentazione. Questo file può essere creato in varie posizioni, a seconda di come sono stati compilati i sorgenti. Se il binario '**`screen`**' è stato previsto con il bit SUID attivo, questo file FIFO potrebbe essere creato nella directory '`/tmp/screens/S-`***utente***`/`', oppure, più utilmente, potrebbe essere creato nella directory '`~/.screen/`'. È da ritenere che questa ultima scelta sia la migliore; volendo, si può utilizzare la variabile di ambiente ***SCREENDIR*** per indicare il percorso della directory che Screen deve usare per i file FIFO.

Il nome utilizzato per il file FIFO serve a identificare una particolare sessione di lavoro di Screen, assieme a tutte le finestre gestite attraverso questa. Di solito, si tratta di un nome articolato secondo il modello seguente:

pid . terminale . nodo

Per esempio, ‘**123.tty4.dinkel**’ è il modo con cui si identifica la sessione di Screen che ha il numero PID 123, utilizza il terminale corrispondente al dispositivo ‘`/dev/tty4`’, sul sistema chiamato ‘**dinkel**’.

Una sessione di Screen, quando è in funzione regolarmente, è ***attaccata*** al terminale fisico che si utilizza effettivamente (questo terminale fisico può anche essere una console virtuale di GNU/Linux). La sessione può essere distaccata e successivamente riattaccata altrove, presso un altro terminale fisico. Le applicazioni in funzione nelle varie finestre di una sessione distaccata, continuano a funzionare regolarmente. Di solito, a meno di modificare la configurazione predefinita, un segnale di aggancio (‘**SIGHUP**’), cosa che generalmente si ottiene disconnettendo la linea attraverso cui è collegato il terminale, provoca solo il distacco della sessione, senza coinvolgere le applicazioni.

Screen può essere controllato attraverso file di configurazione, la cui collocazione può essere varia. Potrebbe trattarsi di ‘`/etc/screenrc`’ per la configurazione globale e di ‘`~/.screenrc`’ per la personalizzazione di ogni utente. Le direttive di questi file non vengono mostrate qui; eventualmente si può consultare la documentazione originale: *screen(1)*.

Screen imposta automaticamente la variabile ***TERM*** al valore ‘**screen**’, in modo da informare opportunamente le applicazioni di adattarsi alle sue caratteristiche.

Quasi tutti i comandi che possono essere impartiti a Screen sono

prefissati dalla combinazione [*Ctrl a*], alla quale segue poi una sequenza di caratteri o di altre combinazioni di tasti, che ovviamente non vengono passati all'applicazione sottostante. Se però si vuole passare proprio la combinazione [*Ctrl a*] all'applicazione, si deve usare la sequenza [*Ctrl a*][*a*].

A volte, Screen ha la necessità di fornire delle indicazioni. Ciò viene fatto sovrascrivendo parte della finestra in uso, di solito nell'ultima riga. Dopo pochi secondi, i messaggi vengono rimossi, ripristinando il testo precedente.

14.13.2 Utilizzo di «screen»

Screen si compone in pratica dell'eseguibile binario '**screen**'. Come accennato in precedenza, viene predisposto spesso in modo da avere il bit SUID attivo e da essere proprietà dell'utente '**root**'. Se non si richiedono funzionalità particolari a questo programma, non è necessaria tale politica.

```
screen [opzioni] [comando [argomenti_del_comando]]
```

Il programma '**screen**' può essere avviato per iniziare una sessione di lavoro attraverso cui gestire delle applicazioni contenute in finestre differenti, oppure per altre funzionalità descritte in occasione della presentazione delle opzioni. Quando si avvia '**screen**' in modo normale, si può aggiungere l'indicazione di un comando (con i suoi argomenti) che si vuole avviare all'interno della prima finestra. Se questo comando non viene specificato, '**screen**' avvia una shell (quella indicata nella variabile di ambiente ***SHELL***, oppure '`/bin/sh`' in sua mancanza).

Quando un programma ospitato all'interno di una finestra di '**screen**' termina di funzionare, la finestra relativa si chiude. Quando una sessione non ha più finestre, termina di funzionare anche il processo '**screen**' relativo.

Tabella 14.95. Alcune opzioni.

Opzione	Descrizione
`-U`	Avvia il programma richiedendo espressamente l'uso della codifica UTF-8 per il terminale.
`-c` ***file***	Permette di specificare un file di configurazione alternativo a quello predefinito.
`-s` ***shell***	Permette di indicare una shell alternativa a quella contenuta nella variabile di ambiente ***SHELL***, la quale viene utilizzata ogni volta che si apre una nuova finestra senza specificare il programma che deve essere avviato al suo interno.
`-S` ***sessione***	Permette di dare un nome a una sessione. A questo nome viene comunque aggiunto il numero PID anteriormente. Lo scopo è quello di rendere più semplice l'identificazione di una sessione.
`-ls` `-list`	Questa opzione va usata da sola: non avvia alcuna nuova sessione e si limita a elencare quelle già aperte dall'utente che ne sta facendo richiesta. Attraverso questo elenco si possono individuare facilmente quali siano le sessioni distaccate, cioè quelle che possono essere riprese utilizzando l'opzione '**-r**'.

Opzione	Descrizione
`-d` [*pid.*] *tty* [*.nodo*] `-D` [*pid.*] *tty* [*.nodo*]	Permette di distaccare una sessione di Screen da un terminale fisico, senza interrompere il funzionamento degli applicativi avviati al suo interno. Si può usare questa opzione assieme a '**`-r`**', in modo da riattaccare la sessione in un altro terminale.
`-r` [[*pid.*] *tty* [*.nodo*]] `-R` [[*pid.*] *tty* [*.nodo*]]	Permette di riattaccare sul terminale in funzione attualmente, una sessione staccata in precedenza. Se non si indica la sessione, viene avviata la prima di quelle che risultano distaccate; se in particolare si utilizza '**`-R`**', si ottiene comunque l'avvio di una sessione anche se non ce ne sono da riprendere. Questa opzione può essere usata da sola o in abbinamento a '**`-d`**' (o '**`-D`**'). In questo ultimo caso, si indica prima l'opzione '**`-d`**', poi '**`-r`**', infine la sessione da staccare e da riattaccare.
`-x` [[*pid.*] *tty* [*.nodo*]]	Questa opzione permette di accedere a una sessione già aperta e funzionante presso un altro terminale fisico. Se non viene specificata la sessione, viene aperta la prima che può essere trovata. Quando si condivide una sessione tra più terminali fisici, ogni terminale può accedere solo alle finestre che non sono attive da qualche parte.

Segue la descrizione di alcuni esempi.

- `$` **`screen`** [*Invio*]

Avvia una sessione di Screen sul terminale da cui si esegue il comando, aprendo la shell predefinita nella prima finestra.

- `$` **`screen -U`** [*Invio*]

  Come nell'esempio precedente, richiedendo espressamente l'uso della codifica UTF-8 per il terminale.

- `$` **`screen mc`** [*Invio*]

  Avvia una sessione di Screen sul terminale da cui si esegue il comando, avviando il programma '**`mc`**', senza argomenti, nella prima finestra.

- `$` **`screen -ls`** [*Invio*]

  Elenca le sessioni aperte dall'utente.

- `$` **`screen -d tty2`** [*Invio*]

  Distacca la sessione in funzione sul terminale identificato dal dispositivo '`/dev/tty2`' (in pratica, la seconda console virtuale). Non vengono indicate altre informazioni per il nome della sessione, perché probabilmente l'informazione del terminale è sufficiente e non crea ambiguità.

- `$` **`screen -d`** [*Invio*]

  Distacca la prima sessione attiva appartenente all'utente stesso.

- `$` **`screen -r tty2`** [*Invio*]

  Attacca, sul terminale da cui si dà il comando, la sessione che in origine è stata avviata sul terminale '`/dev/tty2`' e successivamente distaccata.

- `$` **`screen -r`** [*Invio*]

Attacca la prima sessione libera che trova.

- `$` **`screen -d -r tty2`** [*Invio*]

  Distacca la sessione in funzione sul terminale identificato dal dispositivo '`/dev/tty2`', riattaccandola sul terminale da cui si dà il comando.

- `$` **`screen -d -r`** [*Invio*]

  Distacca la prima sessione attiva che trova e la riattacca sul terminale da cui si dà il comando.

14.13.3 Comandi interattivi

«

Una volta avviato l'eseguibile '**`screen`**', si può interagire con questo attraverso dei comandi composti da combinazioni di tasti. Nella maggior parte dei casi si tratta di sequenze iniziate dalla combinazione [*Ctrl a*].

Per motivi di compatibilità, spesso sono disponibili diversi tipi di sequenze per lo stesso risultato. Nella tabella 14.96 vengono elencate solo alcune di queste sequenze; per un elenco completo occorre leggere la documentazione originale, costituita dalla pagina di manuale: *screen(1)*.

Tabella 14.96. Alcuni dei comandi che si possono dare a Screen, quando è in funzione.

Sequenza	Effetto
[*Ctrl a*][*?*]	Mostra una guida rapida ai comandi disponibili.
[*Ctrl a*][*a*]	Invia la combinazione [*Ctrl a*] all'applicazione attiva.
[*Ctrl a*][***n***]	Seleziona l'***n***-esima finestra. La prima ha il numero zero.
[*Ctrl a*][*n*]	Passa alla finestra successiva.

Sequenza	Effetto
[*Ctrl a*][*p*]	Passa alla finestra precedente.
[*Ctrl a*][*c*]	Crea una nuova finestra.
[*Ctrl a*][*d*]	Distacca la sessione dal terminale fisico.
[*Ctrl a*][*w*]	Mostra un breve riepilogo delle finestre esistenti.
[*Ctrl a*][*Esc*]	Inizia la modalità di scorrimento e copia all'indietro.
[*Ctrl a*][*]*]	Incolla il testo inserito precedentemente nella memoria tampone.

Le operazioni più complesse sono quelle che riguardano la copia e l'inserimento di testo che proviene da quanto visualizzato attualmente, o nel testo precedente. Infatti, per ogni finestra viene conservato uno storico delle righe visualizzate, il quale può essere rivisto e dal quale si possono prelevare delle parti, inserendole in una memoria tampone (la documentazione *screen(1)* parla di *paste buffer*).

Con il comando [*Ctrl a*][*Esc*] si inizia la modalità di scorrimento e copia, cosa che blocca il funzionamento dell'applicazione che utilizza la finestra attiva. Da quel momento, si possono usare i tasti freccia e pagina per spostare il cursore; eventualmente si possono usare i tasti [*h*], [*j*], [*k*] e [*l*], come si fa con VI (sezione 22.15). Si possono anche fare delle ricerche nello stile di VI, con i comandi [*/*] e [*?*].

Quando si raggiunge il pezzo che si vuole copiare nella memoria tampone, lo si deve delimitare. Ciò si ottiene normalmente premendo il tasto [*barra−spaziatrice*] nel punto di inizio, quindi si fa scorrere il cursore nel punto finale e si preme nuovamente la [*barra−spaziatrice*] per concludere. La selezione del testo coincide anche con la conclusione della modalità di scorrimento e copia,

cosa che dopo poco fa riprendere il funzionamento del programma.

È possibile anche la selezione di testo in modo rettangolare. Per questo, dopo aver premuto la [*barra−spaziatrice*] per indicare il punto di inizio, si deve aggiungere anche il tasto [*c*], a indicare un bordo sinistro, oppure [*C*] a indicare un bordo destro. Successivamente, quando si raggiunge anche il punto finale, si preme nuovamente [*C*], oppure [*c*] (a seconda di come si è iniziato) prima della [*barra−spaziatrice*].

Infine, il comando [*Ctrl a*][*]] inserisce il testo, accumulato precedentemente nella memoria tampone, nello standard input dell'applicazione contenuta nella finestra attiva.

14.13.4 Configurazione

«

Il programma Screen consente di definire la sua configurazione attraverso i file '`/etc/screenrc`' e '`~/.screenrc`' (il primo per tutto il sistema; il secondo per ogni utente). In alternativa, attraverso l'opzione '**`-c`**' si può indicare un file differente.

Il contenuto del file di configurazione si compone di commenti, preceduti dal simbolo '**`#`**', righe vuote, righe bianche e direttive. Le informazioni utili sono costituite soltanto dalle direttive (istruzioni semplici), disposte normalmente su una sola riga.

14.14 Console parallele con Pconsole

«

A ogni terminale a caratteri gestito presso il proprio elaboratore, corrisponde un file di dispositivo, individuabile normalmente con il programma '**`tty`**':

```
$ tty [Invio]
```

```
/dev/tty5
```

Il risultato del comando '**tty**' è relativo all'elaboratore presso il quale si sta operando, pertanto, se si tratta di un accesso remoto, il file di dispositivo riportato è inteso essere quello corrispondente nell'elaboratore remoto. Si cerchi di seguire l'esempio seguente, dove dall'elaboratore «A» ci si collega all'elaboratore «B», con l'aiuto di un programma appropriato:

```
A$ tty [Invio]
```

Ci si trova seduti davanti alla console dell'elaboratore «A» e si controlla quale file di dispositivo corrisponde alla propria console; si suppone si tratti della quinta console virtuale di un sistema GNU/Linux:

```
/dev/tty5
```

Con l'aiuto di un programma appropriato, ci si collega all'elaboratore «B» (che ha indirizzo IPv4 172.17.1.2):

```
A$ ssh 172.21.1.2 [Invio]

Password: digitazione_all'oscuro [Invio]

 Last login: … from …
B$
```

A questo punto, pur essendo seduti davanti alla console dell'elaboratore «A», si sta lavorando nell'elaboratore «B». Si verifica il file di dispositivo corrispondente al terminale usato:

```
B$ tty [Invio]
```

```
/dev/pts/2
```

Nonostante questo, rimane il fatto che il terminale usato presso l'elaboratore «A», ovvero quello locale, è '`/dev/tty5`'; attraverso tale terminale si accede all'elaboratore remoto «B»; presso l'elaboratore «B» il terminale usato è '`/dev/pts/2`'.

Se presso l'elaboratore «A» un programma fosse in grado di controllare il file di dispositivo '`/dev/tty5`', questo controllerebbe in pratica l'accesso remoto all'elaboratore «B», secondo la simulazione dell'esempio.

Questa premessa è necessaria per comprendere il funzionamento del programma '**`pconsole`**',[20] con il quale è possibile inviare un comando simultaneamente a più terminali, utilizzando i file di dispositivo **locali**:

```
pconsole
```

Il programma '**`pconsole`**' si usa senza argomenti, con i privilegi dell'utente '**`root`**', ma dopo l'avvio gli si devono impartire dei comandi:

```
# pconsole [Invio]

  pconsole WJ101

  pconsole command mode
>>>
```

Con il comando '**help**' è possibile ottenere il riepilogo dei comandi disponibili:

```
>>> help [Invio]

   help            Give help about the available commands
   ?               short-cut for 'help'
   version         Display version information
   echo            Turn echo on or off
   attach          Attach to a tty device
   detach          Detach from a tty device
   list            Show devices currently attached to
   connect         Leave command mode
   quit            Exit pconsole
   exit            Exit pconsole

>>> quit [Invio]
```

Il programma ha due stati di funzionamento: la modalità di comando, corrispondente a quella mostrata negli esempi; la modalità di invio. La modalità di comando serve, evidentemente, per impartire dei comandi, mentre l'altra modalità consente di passare all'inserimento di testo da inviare ai terminali che risultano attaccati. In pratica, prima ci si attacca a dei terminali, ovvero ai file di dispositivo corrispondenti, quindi si passa in modalità di inserimento e ciò che si scrive, viene eseguito in tutti i terminali relativi; quello che non si può fare attraverso '**pconsole**' è di «vedere» ciò che accade presso i vari terminali.

Per passare alla modalità di comando, si usa il codice <*SOH*>, ovvero <^*a*>, il quale si ottiene normalmente con la combinazione [*Ctrl a*]; per passare alla modalità di invio, si usa il codice <*EOT*>, ovvero <^*d*>, il quale si ottiene normalmente con la combinazione [*Ctrl d*].

È abbastanza difficile mostrare un esempio completo di utilizzo del programma '**pconsole**'; per capire ciò che si vuole sintetizzare nei comandi seguenti richiede una buona dose di intuito. Per cominciare, si suppone di avere aperto diversi terminali; in particolare interessano la console virtuale corrispondente al file di dispositivo '`/dev/tty3`' e un terminale grafico corrispondente al file di dispositivo '`/dev/pts/4`'. Presso un altro terminale libero (una console virtuale o un terminale grafico, senza che ciò faccia differenza) si avvia '**pconsole**' e ci si «attacca» ai due terminali già nominati:

```
# pconsole[Invio]

  pconsole WJ101

  pconsole command mode
>>> attach /dev/tty3[Invio]

  attaching /dev/tty3 : Ok
>>> attach /dev/pts/4[Invio]

  attaching /dev/pts/4 : Ok
```

Con il comando '**list**' si può vedere quali terminali risultano attaccati:

```
>>> list[Invio]

  Currently attached to:
   /dev/tty3                              (device no 4, 3)
   /dev/pts/4                             (device no 136, 4)
```

Si decide di passare alla modalità di inserimento:

```
>>> [Ctrl d]
```

```
  Press <Ctrl-A> for command mode

>
```

Come si può osservare, l'invito assume una forma più breve, per ricordare che ci si trova nella modalità di inserimento. Da qui si vuole semplicemente impartire il comando '**ls**', che viene eseguito in pratica nei due terminali controllati da '**pconsole**':

```
> ls [Invio]
```

Dal terminale dove si sta usando '**pconsole**', non si vede alcun risultato; per sapere cosa è successo effettivamente, occorre invece passare agli altri terminali.

Al termine, si torna alla modalità di comando:

```
> [Ctrl a]

  pconsole command mode

>>>
```

A questo punto si può chiudere:

```
>>> quit [Invio]

  detaching from /dev/tty3 : Ok
  detaching from /dev/pts/4 : Ok
```

Per concludere, c'è da osservare che la documentazione di Pconsole suggerisce di attribuire al programma i permessi SUID-root, per consentire a tutti gli utenti di usarlo; tuttavia, ciò è sicuramente sconsigliabile per motivi di sicurezza.

14.15 Getty

«

Nella procedura di inizializzazione del sistema, Getty è quel programma che si occupa di attivare il terminale e iniziare la procedura di accesso. Come dice la pagina di manuale *getty(1)*: «Getty è il secondo dei tre programmi (*init(1)*, *getty(1)* e *login(1)*) utilizzati dal sistema per permettere all'utente di accedere». In pratica, il programma Getty si occupa di:

- aprire la linea di terminale e impostare le modalità necessarie;
- emettere l'invito della procedura di accesso;
- ricevere il nominativo usato dall'utente per identificarsi;
- attivare il programma per la procedura di accesso (convenzionalmente si tratta di '`/bin/login`'), fornendogli già il nominativo-utente (successivamente è compito di '**`login`**' la richiesta di inserire la parola d'ordine).

Il programma Getty tipico fa uso di alcuni file:

- '`/etc/gettydefs`'
  per la definizione delle caratteristiche delle linee dei terminali;
- '`/etc/issue`'
  per definire un testo di «benvenuto» da inviare all'utente che tenta di connettersi.

14.15.1 Utilizzo di un programma Getty

Un programma Getty non è fatto per l'utilizzo manuale diretto, ma per essere inserito nel file '`/etc/inittab`', in modo da essere attivato direttamente da Init durante la fase di inizializzazione del sistema. In un sistema GNU/Linux, l'attivazione delle sei console virtuali consuete avviene con record simili a quelli seguenti:

```
1:12345:respawn:/sbin/getty tty1
2:2345:respawn:/sbin/getty tty2
3:2345:respawn:/sbin/getty tty3
4:2345:respawn:/sbin/getty tty4
5:2345:respawn:/sbin/getty tty5
6:2345:respawn:/sbin/getty tty6
```

Come si vede dall'esempio, viene usato un argomento per specificare il terminale da utilizzare, ovvero il nome del file di dispositivo corrispondente contenuto nella directory '`/dev/`'. Questo elemento, viene definito normalmente come «linea», alludendo al tipo di terminale in base al tipo di connessione utilizzata.

Quando il programma Getty viene utilizzato per attivare una connessione attraverso un terminale seriale, si pone il problema di configurare opportunamente la porta seriale stessa. In tal caso si utilizzano altri argomenti, oppure la configurazione del file '`/etc/gettydefs`'.

Se oltre alla linea seriale si utilizzano dei modem, si aggiunge anche il problema della loro inizializzazione. Il programma Getty può solo occuparsi di quello connesso dalla sua parte, ma anche in tal caso si pone il problema di definire la stringa di inizializzazione adatta.

Quando si vuole ottenere una connessione attraverso modem, utilizzando una linea telefonica commutata, Getty deve essere in grado di

controllare il modem anche in questo modo, rispondendo e distinguendo eventualmente se la chiamata proviene da un altro modem o se si tratta di un segnale sonoro normale.

14.15.2 File «/etc/issue»

«

In generale, i programmi Getty utilizzano il file '`/etc/issue`' per definire il messaggio introduttivo di invito della procedura di accesso. Nel testo che compone questo file possono essere utilizzati alcuni codici di escape per ottenere effetti particolari. Questi codici dipendono dall'interpretazione del programma Getty che li utilizza. In particolare, l'elenco della tabella 14.111 è adatto sia a Agetty, sia a MinGetty.

Tabella 14.111. Elenco dei codici che '**agetty**' e '**mingetty**' riconoscono nel file '`/etc/issue`'.

Codice	Descrizione
`\b`	Inserisce la velocità della linea utilizzata.
`\d`	Inserisce la data.
`\s`	Inserisce il nome del sistema operativo.
`\l`	Inserisce il nome della linea di terminale utilizzata.
`\m`	Inserisce il nome dell'architettura della macchina.
`\n`	Inserisce il nome dell'elaboratore: *hostname*.
`\o`	Inserisce il nome a dominio dell'elaboratore.
`\r`	Inserisce il numero di rilascio del sistema operativo.

Codice	Descrizione
`\t`	Inserisce l'orario.
`\u`	Inserisce il numero di utenti connessi.
`\U`	Come '**`\u`**', ma aggiunge la parola '**`user`**' o '**`users`**'.
`\v`	Inserisce la versione del sistema operativo.

Tuttavia va considerato che esistono differenze molto grandi tra i vari programmi Getty per i codici di escape utilizzabili nel file '`/etc/issue`'; pertanto, l'unico modo per predisporne una versione standard unificata, è quello di fare a meno di questi. Alcune distribuzioni GNU/Linux, a questo proposito, predispongono il file '`/etc/issue`' attraverso la procedura di inizializzazione del sistema.

14.15.3 Utilizzo di MinGetty

MinGetty[21] è un programma Getty minimo, per l'accesso esclusivo attraverso console virtuali di GNU/Linux. Per questo, è indicato particolarmente per risparmiare memoria nei sistemi minimi e non richiede file di configurazione, a parte il messaggio di pubblicazione nel file '`/etc/issue`'. È composto da un solo file eseguibile, che potrebbe avere il nome '**`mingetty`**', o '**`getty`**'.

`mingetty` [***opzioni***] ***console_virtuale***

`getty` [***opzioni***] ***console_virtuale***

Tabella 14.112. Alcune opzioni.

Opzione	Descrizione
`--noclear`	Non ripulisce lo schermo prima di avviare la procedura di accesso.
`--long-hostname`	Visualizza il nome completo dell'elaboratore all'atto dell'attivazione della procedura di accesso.

L'esempio seguente si riferisce a record del file '`/etc/inittab`':

```
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

Questi record attivano le prime sei console virtuali in corrispondenza dei livelli di esecuzione da due a cinque. In particolare, la prima console virtuale viene attivata anche con il livello uno.

14.15.4 Utilizzo di Agetty

«

Agetty,[22] ovvero *Alternative Linux getty*, è un programma Getty ridotto, per l'accesso attraverso console virtuali di GNU/Linux e le porte seriali. Non richiede file di configurazione, a parte il messaggio di pubblicazione nel file '`/etc/issue`', mentre le impostazioni eventuali vanno date attraverso la riga di comando.

`agetty` [***opzioni***] ***velocità*** [`,` …] ***porta*** [***variabile_term***]

```
getty [opzioni] velocità [,...] porta [variabile_term]
```

Tabella 14.114. Argomenti e alcune opzioni valide con il programma '**agetty**'.

Argomento od opzione	Descrizione
porta	Si tratta del nome del file di dispositivo che identifica il terminale da utilizzare. È riferito alla directory '`/dev/`', quindi, per esempio, '**`tty1`**' si riferisce al file di dispositivo '`/dev/tty1`'.
velocità	È un elenco, separato da virgole, di una o più velocità di trasmissione espresse in bit al secondo (simbolo: «bit/s»). Ogni volta che il programma riceve un carattere *break*, seleziona la velocità successiva nell'elenco. Dopo l'ultima, riprende dalla prima. Di solito, si preferisce indicare le velocità in ordine decrescente.
variabile_term	L'ultimo argomento può essere il valore da assegnare alla variabile ***TERM*** che poi viene ereditata da '**`login`**' e dalla shell.
`-h`	Abilita il controllo di flusso hardware (RTS/CTS).

Argomento od opzione	Descrizione
`-i`	Non visualizza il contenuto del file '`/etc/issue`' (e di nessun altro equivalente) prima di emettere l'invito della procedura di accesso. Ciò può essere utile nel caso in cui, per motivi tecnici, sia preferibile evitare di inviare troppi dati prima della richiesta di identificazione che introduce la procedura di accesso.
`-f` ***file***	Permette di indicare un nome diverso da quello predefinito per il file contenente il messaggio di pubblicazione (quello che di solito è '`/etc/issue`').
`-I` ***stringa_di_inizializzazione***	Permette di inviare al terminale, o al modem, una stringa di inizializzazione, prima di qualunque altro dato. Per indicare caratteri speciali non stampabili, si può usare la forma ottale di tre cifre numeriche precedute da una barra obliqua inversa ('****').
`-l` ***programma_di_login***	Permette di indicare un programma per la procedura di accesso diverso da quello predefinito: '`/bin/login`'.
`-m`	Fa in modo che il programma tenti di determinare la velocità da utilizzare dal messaggio di stato '**`CONNECT`**' dei modem compatibili Hayes.
`-n`	Fa in modo che non venga richiesta l'identificazione attraverso la procedura di accesso. Può avere senso questa modalità se si utilizza anche l'opzione '**`-l`**' per accedere al sistema in modo diverso.

Argomento od opzione	Descrizione
`-t` ***tempo_massimo***	Permette di definire un tempo massimo di attesa, espresso in secondi. Se l'accesso non viene effettuato entro il tempo previsto, si interrompe la comunicazione. Di solito si utilizza questa opzione solo per le connessioni remote attraverso l'uso del modem.
`-L`	Specifica in modo esplicito che si tratta di una linea locale, senza che ci sia la necessità di individuare la presenza di una portante. Può essere utile, se si utilizza una linea seriale locale che non dispone del segnale di portante.
`-w`	Attende di ricevere dall'utente o dal modem un segnale di <*CR*> o <*LF*>, prima di inviare il messaggio di pubblicazione (di solito si tratta del contenuto del file '`/etc/issue`') e la richiesta di identificazione per la procedura di accesso. L'uso di questa opzione è molto importante se si utilizza anche l'opzione '**`-I`**'.

Segue la descrizione di alcuni esempi.

- `1:12345:respawn:/sbin/getty 38400 tty1`

  Attiva la prima console virtuale in un sistema GNU/Linux.

- `1:23:respawn:/sbin/getty -L 9600 ttyS0 vt100`

  Attiva un terminale locale, connesso direttamente attraverso la prima porta seriale. Viene anche indicato un tipo di terminale pari alla sigla '**`vt100`**'.

- `1:23:respawn:/sbin/getty -mt60 ttyS0 9600,2400,1200`

  Attiva un terminale remoto, collegato attraverso un modem alla prima porta seriale, utilizzando diverse velocità alternative.

14.15.5 Annotazione per la predisposizione di un terminale seriale

«

Un terminale seriale può essere predisposto semplicemente utilizzando un elaboratore con un sistema operativo qualunque, purché provvisto di software necessario a riprodurre il comportamento di un terminale seriale. Per fare un esempio con lo stesso sistema GNU/Linux, si può utilizzare il programma Minicom.

In linea di massima, i vari programmi Getty sono predisposti per la consuetudine diffusa di usare una codifica «8N1», ovvero: 8 bit dati senza alcuna parità e un bit di stop. Questa impostazione va mantenuta sempre, a meno di sapere esattamente quello che si sta facendo.

Il parametro più importante che può essere modificato è la velocità espressa in bit al secondo, la quale deve essere stabilita precisamente e in modo identico tra Getty e il programma di emulazione di terminale all'altro capo del filo.

Il controllo di flusso dovrebbe essere sempre di tipo hardware, cioè RTS/CTS. Eccezionalmente si può usare un controllo di flusso software, cioè XON/XOFF, per esempio quando si è costretti a utilizzare un cavo seriale a tre fili; ciò purché Getty sia in grado di operare in questo modo e che la velocità di comunicazione sia sufficientemente bassa da permetterlo (da 9600 bit/s in giù).

Come ultimo problema, occorre verificare per quali tipi di terminali standard può essere configurato il programma di emulazione. Generalmente, tutti i programmi di questo tipo dovrebbero essere in grado di operare come terminali VT100; se però il programma a disposizione offre di meglio, è sempre il caso di sfruttare tali caratteristiche.

L'uso di un terminale seriale rispetto a una console comporta delle conseguenze operative non trascurabili, legate al comportamento della tastiera e alla visualizzazione dei caratteri sul video.

In pratica, la visualizzazione di certi simboli può variare, specialmente le bordature vengono sostituite con caratteri normali; inoltre, alcuni tasti funzionali e alcune combinazioni di tasti diventano inutilizzabili.

14.16 Console nei sistemi GNU/Linux

Nei sistemi Unix, la console è il terminale principale e come tale ha un ruolo fondamentale. Generalmente, con GNU/Linux non si avverte questo particolare perché la gestione normale delle console virtuali fa sì che la «console» sia semplicemente quella console virtuale che si adopera in un momento determinato. Per fare un po' di chiarezza tra console e console virtuali, è bene dare un'occhiata ai file di dispositivo.

```
# ls -l /dev/console /dev/tty /dev/tty[0-9] [Invio]
```

Il risultato dovrebbe essere quello che segue, tenendo conto che la proprietà e i permessi cambiano in funzione dell'uso che si sta facendo in un momento determinato:

```
crw-------  1 root   root   5,   1 mag  5  1998 /dev/console
crw-rw-rw-  1 root   root   5,   0 mag  5  1998 /dev/tty
crw-------  1 root   root   4,   0 mag  5  1998 /dev/tty0
crw-------  1 root   root   4,   1 mag  5  1998 /dev/tty1
crw-------  1 root   root   4,   2 mag  5  1998 /dev/tty2
crw-------  1 root   root   4,   3 mag  5  1998 /dev/tty3
crw-------  1 root   root   4,   4 mag  5  1998 /dev/tty4
crw-------  1 root   root   4,   5 mag  5  1998 /dev/tty5
crw-------  1 root   root   4,   6 mag  5  1998 /dev/tty6
crw-------  1 root   root   4,   7 mag  5  1998 /dev/tty7
crw-------  1 root   root   4,   8 mag  5  1998 /dev/tty8
crw-------  1 root   root   4,   9 mag  5  1998 /dev/tty9
```

Si osservi prima di tutto che il dispositivo ‘`/dev/console`’ ha numero primario cinque e numero secondario uno, mentre in origine si utilizzavano i numeri 4,0, corrispondenti al dispositivo ‘`/dev/tty0`’. Se dovesse essere necessario, si possono creare questi due file di dispositivo con i comandi seguenti (si comincia dalla cancellazione di quelli vecchi).

```
# rm /dev/console /dev/tty0 [Invio]

# mknod -m 600 /dev/console c 5 1 [Invio]

# mknod -m 600 /dev/tty0 c 4 0 [Invio]
```

Osservando i numeri primario e secondario dell’elenco mostrato, si comprende meglio lo scopo di questi file di dispositivo. I file di dispositivo ‘`/dev/tty1`’, ‘`/dev/tty2`’,... rappresentano ognuno una console virtuale; il file di dispositivo ‘`/dev/tty0`’ rappresenta quella attiva, mentre ‘`/dev/tty`’ rappresenta il terminale attivo, in senso più ampio.

La console è il terminale principale di un sistema, quello su cui devo-

no apparire i messaggi di sistema più importanti e quello che viene usato dai programmi in mancanza d'altro. Quando GNU/Linux poteva gestire esclusivamente console rappresentate dalla tastiera dell'elaboratore e dall'adattatore grafico tradizionale, era corretto a considerare '`/dev/console`' un modo alternativo di identificare la console virtuale attiva; ma all'estendersi delle possibilità di GNU/Linux, diventa importante poter definire espressamente a cosa corrisponda tale dispositivo.

14.16.1 Definizione esplicita della console

Se non viene specificato diversamente, la console, cioè il dispositivo '`/dev/console`', corrisponde semplicemente alla prima unità in grado di assolvere allo scopo; nella maggior parte dei casi si tratta della console virtuale attiva in un certo momento.

Non esistendo un'unità fisica corrispondente univocamente alla console, questa può essere soltanto associata a un altro dispositivo esistente, come una console virtuale o un altro tipo di terminale. Con i kernel ≥ 2.2.∗ è possibile abbinare la console alla console virtuale attiva, a una console virtuale specifica o a una linea seriale. Per questo si interviene con un parametro del kernel.

```
console=dispositivo [ , opzioni ]
```

Al parametro '**`console`**' può essere abbinato il dispositivo a cui si vuole fare riferimento, senza aggiungere il percorso ('`/dev/`') e, se necessario, possono essere aggiunte altre opzioni che riguardano la velocità, la parità e il numero di bit. Per esempio, il parametro '**`console=tty10`**', fa in modo che la decima console virtuale sia

anche la console vera e propria.

Utilizzando il parametro '**console**', si stabilisce a cosa corrisponde il dispositivo '`/dev/console`'. Dipende dai programmi il fatto che tale dispositivo venga utilizzato o meno. In generale, questo significa che i messaggi più importanti appaiono lì; niente di più.

Se si avvia il kernel attraverso SYSLINUX, il parametro può essere fornito all'interno della direttiva '**append**', come si vede nell'esempio seguente:

```
LABEL linux
 KERNEL vmlinuz
 APPEND root=/dev/sda4 ro console=tty10
```

Con GRUB 1 si può inserire il parametro in coda all'istruzione '**kernel**':

```
kernel (hd0,5)/boot/vmlinuz-2.4.2 root=/dev/hda6 console=tty12 ro
```

14.16.2 Usare o non usare la console

«

È bene ribadire che la console è sempre ospitata da un altro terminale identificato in modo più preciso. Generalmente, '`/dev/console`' serve solo per avere un riferimento: il dispositivo a cui mandare i messaggi più importanti, contando che questi siano letti dall'interessato. In pratica, stando così le cose, il dispositivo '`/dev/console`' viene aperto sempre solo in scrittura per visualizzare qualcosa e mai, o quasi, per ricevere un inserimento dati da tastiera. Se poi la console corrisponde a un terminale su cui si sta lavorando normalmente, i messaggi diretti a questa servono per disturbare l'utente confondendogli il contenuto dello schermo.

Per poter interagire con un terminale qualunque, di solito si interviene nel file ‘/etc/inittab’, specificando l’avvio di un programma Getty abbinato a un dispositivo di terminale determinato. Si osservi l’esempio:

```
…
1:12345:respawn:/sbin/getty tty1
2:2345:respawn:/sbin/getty tty2
3:2345:respawn:/sbin/getty tty3
4:2345:respawn:/sbin/getty tty4
5:2345:respawn:/sbin/getty tty5
6:2345:respawn:/sbin/getty tty6
…
```

Questo è già stato descritto nel capitolo precedente: si tratta dell’attivazione delle prime sei console virtuali, in modo che da quelle possa essere eseguito l’accesso. Tutte le altre console virtuali esistono ugualmente, solo che da quelle non si può fare nulla, a meno di «scriverci» inviando dei messaggi, oppure di utilizzare un programma che ci faccia qualcosa d’altro.

Se la console vera e propria viene abbinata a una console virtuale «libera», quello che si ottiene è di mandare lì i messaggi diretti alla console, così da non disturbare l’utente che sta usando una console virtuale; ma per il momento, questo non significa che la console venga utilizzata anche per accedere. Ma allora, si può accedere attraverso ‘/dev/console’? Sì, solo che non conviene, perché la console è sempre ospite di un altro tipo di terminale, per cui è meglio attivare un accesso su quel terminale, piuttosto che sulla console generica.

A titolo di esempio, ribadendo che non si tratta di una buona idea, si elencano i passi necessari per poter attivare un accesso su ‘/dev/

console’:

1. deve essere definito in modo esplicito a cosa corrisponde la console attraverso il parametro del kernel ‘**console**’;
2. deve essere aggiunta una riga adatta nel file ‘/etc/inittab’ per l’avvio di un programma Getty che utilizzi il dispositivo ‘/dev/console’;
3. deve essere rimosso il file ‘/etc/ioctl.save’ generato da Init, in quanto contiene l’impostazione iniziale di ‘**stty**’ che la prima volta potrebbe essere incompatibile con le caratteristiche della connessione seriale.

Per definire che la console è abbinata a un dispositivo di terminale determinato, si può utilizzare il parametro del kernel ‘**console**’, come è già stato mostrato; per l’attivazione del programma Getty si può aggiungere la riga seguente al file ‘/etc/inittab’.

```
7:12345:respawn:/sbin/getty -L 19200 console
```

Viene utilizzato proprio il programma ‘**getty**’, con delle opzioni di compromesso, in modo da poter funzionare sia su una console virtuale di GNU/Linux, sia su un terminale seriale.

L’unico vantaggio di agire in questo modo, potrebbe essere quello di consentire l’avvio del sistema stabilendo di volta in volta quale sia la console attraverso un parametro del kernel.

14.16.3 Annotazioni per una console su un terminale seriale

«

Prima di poter attivare una console su un terminale seriale occorre essere in grado di attivare un terminale seriale normale. L’argomento

può essere prematuro, ma serve per completare la discussione sulle problematiche riferite all'uso della console.

Per la gestione di una console su un terminale seriale occorre che il kernel sia stato predisposto per questo: sia per la gestione delle porte seriali, sia la gestione della console su terminale seriale (sezione 8.3.8.9).

L'abbinamento della console a un terminale seriale non ha nulla di complicato: basta utilizzare il parametro '**`console`**', indicare il dispositivo seriale opportuno e la velocità di trasmissione. Gli esempi seguenti sono equivalenti.

'**`console=ttyS1,9600`**'

'**`console=ttyS1,9600n8`**'

L'opzione '**`9600n8`**' rappresenta la velocità a 9600 bit/s, l'assenza di parità ('**`n`**') e la dimensione a 8 bit. In particolare, la parità potrebbe essere espressa attraverso altre lettere:

- '**`n`**' nessuna parità;
- '**`o`**' dispari (*odd*);
- '**`e`**' pari (*even*).

Questo basta a fare in modo che il terminale (configurato opportunamente secondo le stesse caratteristiche) connesso alla porta seriale specificata (nell'esempio è '`/dev/ttyS1`', cioè la seconda porta seriale) sia in grado di funzionare in qualità di '`/dev/console`'.

Le caratteristiche della connessione seriale che possono essere configurate sono molto poche. In particolare, è importante osservare che si sottintende un controllo di flusso hardware (RTS/CTS), per cui il cavo seriale utilizzato deve essere completo.

Se si vuole fare qualcosa di più della semplice visualizzazione dei messaggi emessi e destinati alla console, è necessario attivare un programma Getty, ma in tal caso bisogna stabilire se si vuole fare riferimento al terminale seriale effettivo, o alla console generica. Qualunque sia la scelta, si deve intervenire nel file '`/etc/inittab`'.

```
7:12345:respawn:/sbin/getty -L 9600 ttyS1 vt100
```

Quella che si vede sopra è la riga necessaria ad attivare direttamente il terminale connesso alla seconda porta seriale; l'esempio successivo riguarda invece la console generica.

```
7:12345:respawn:/sbin/getty -L 9600 console
```

Se la console seriale deve poter sostituire completamente il video e la tastiera dell'elaboratore, è necessario rendere consapevole di questo anche il sistema di avvio di GNU/Linux, in modo che l'invito di avvio (il *bootprompt*) appaia sul terminale giusto. GRUB 1 è in grado di farlo attraverso le direttive seguenti:

```
serial --unit=n_porta_seriale --speed=velocità [--dumb]
```

In questo modo si può definire la porta seriale e la velocità, dove rimane implicito il fatto che si usino byte interi senza parità. In particolare, GRUB 1 prevede che si tratti di un terminale compatibile con lo standard VT100; se le cose non fossero così, si può inserire

l'opzione '**--dumb**', come si vede nel modello sintattico.

Tuttavia, la direttiva indicata serve solo a definire l'esistenza di un terminale aggiuntivo, attraverso la porta seriale. Per attivare effettivamente il suo utilizzo, è necessaria la direttiva '**terminal**':

```
terminal serial [console]
```

Con '**terminal serial**' si stabilisce l'uso del terminale seriale come console per l'avvio; se si aggiunge anche la parola chiave '**console**', si fa in modo di consentire la scelta: è sufficiente premere un tasto sul terminale per selezionare implicitamente la console ai fini dell'uso di GRUB 1. Segue un esempio completo del file di configurazione di GRUB 1:

```
default 0
timeout 5

title Console su terminale seriale
kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 console=ttyS1,9600 ro
serial --unit=1 --speed=9600
terminal serial

title Console normale
kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 ro
```

Prima di provare l'uso di una console seriale, occorre essere certi che il terminale seriale funzioni, attraverso programmi come Minicom, anche attivando semplicemente il terminale senza attribuirgli il livello di console. Infine, è importante cancellare il file '`/etc/ioctl.save`' prima di provare.

14.17 Terminali per non vedenti

«

Il sistema di scrittura per non vedenti è stato inventato da Louis Braille nel 1829, da cui viene il nome. Luis Braille era francese e in questo senso vanno viste le scelte che possono sembrare insolite, per chi è abituato a convenzioni derivanti dalla lingua inglese. Il sistema codifica 63 simboli rappresentati ognuno da una ***cella*** in cui possono essere collocati un massimo di sei punti in rilievo:

```
.----.
|*  *|
|*  *|
|*  *|
`----'
```

Tanto per fare un esempio, se si possono immaginare gli asterischi della figura seguente come altrettanti punti in rilievo, la parola «ciao!» si può rappresentare in questo modo:

```
* *   * *   *
  *       * * *
        * *
```

Il sistema braille è fatto fondamentalmente per una scrittura manuale, attraverso l'uso di una tavoletta speciale, come si può vedere nelle figure 14.128 e 14.129.

Figura 14.128. Tavoletta di scrittura braille normale.

La scrittura avviene su un foglio di cartoncino bloccato sulla tavoletta, con l'aiuto di un punteruolo, guidato dalle feritoie poste su un regolo mobile (nel caso della tavoletta tascabile, il regolo è fisso). Dal momento che la scrittura avviene attraverso l'incisione del cartoncino, la lettura tattile, in rilievo, può avvenire solo sull'altro lato; pertanto, per quanto riguarda le lingue latine, la scrittura avviene da destra verso sinistra e la lettura da sinistra a destra.

Figura 14.129. Tavoletta di scrittura braille tascabile. La fase si scrittura avviene incidendo il cartoncino. In questo caso, non si vede la fila superiore, perché rimane coperta dall'ombra.

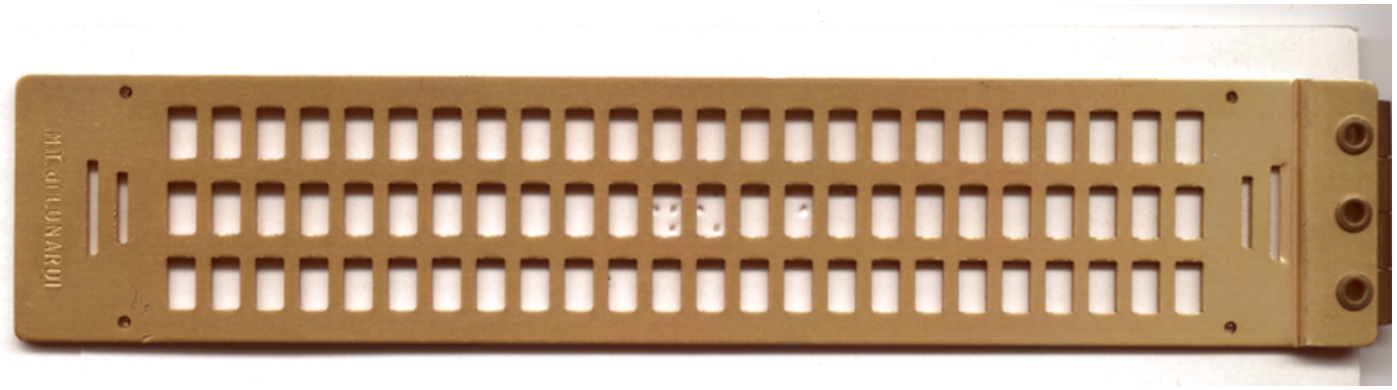


Figura 14.130. Testo inciso da destra verso sinistra.

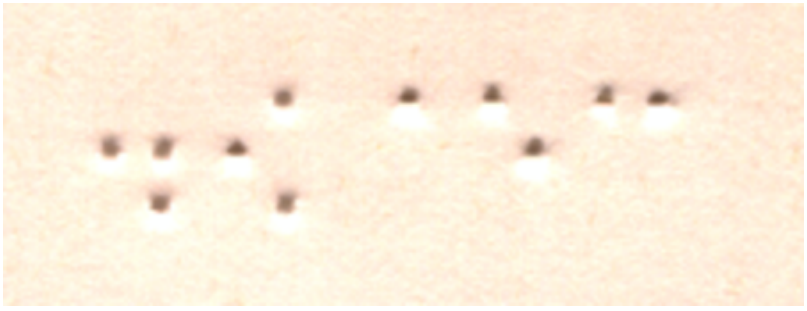
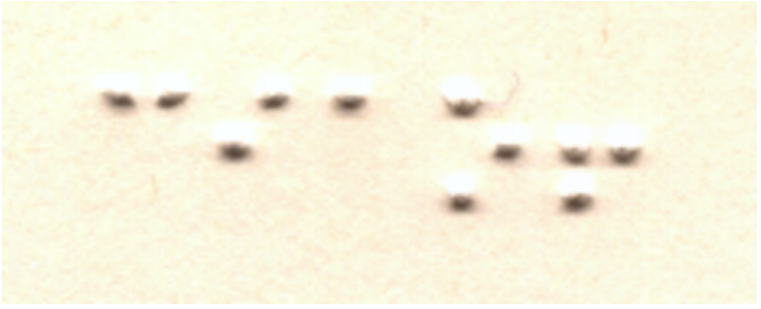


Figura 14.131. Testo in rilievo da sinistra verso destra.



Il braille standard è composto da celle con un massimo di sei punti in rilievo, per un massimo di 2^6-1 segni, più lo spazio; tuttavia, possono esistere anche dispositivi braille predisposti per un massimo di otto punti in rilievo.

14.17.1 Codifica semplificata

La disposizione classica della codifica del braille è sintetizzata dalle figure 14.134 e 14.135. Chi volesse studiare ed esercitarsi con il braille, farebbe bene a copiare le due figure in modo da mettere a sinistra quella dei segni in scrittura e a destra quella dei segni in lettura.

Come si può comprendere a livello intuitivo, i 63 simboli non consentono di distinguere le lettere minuscole da quelle maiuscole; in generale si tratta sempre solo di lettere minuscole che diventano maiuscole se precedute dal segno relativo. Per esempio, «Daniele» si scrive così:

```
*   * * *     * *   * *   *     *
      *         * *     * *       *
*             *             *
```

Nello stesso modo, come si vede, non sono previsti i numeri. Questi si ottengono indicando il segno di numero davanti alle prime 10 lettere alfabetiche. Per esempio, 123 si indica così:

```
  *   *     *     * *
  *         *
* *
```

Pertanto, a differenza delle maiuscole, il segno di numero vale per tutta la sequenza di simboli successiva.

Esistono naturalmente altri dettagli molto importanti per chi deve usare il braille, ma qui non vengono descritti. Purtroppo, i segni a disposizione sono molto pochi e di conseguenza, tutto quanto ha sempre un valore relativo al contesto.

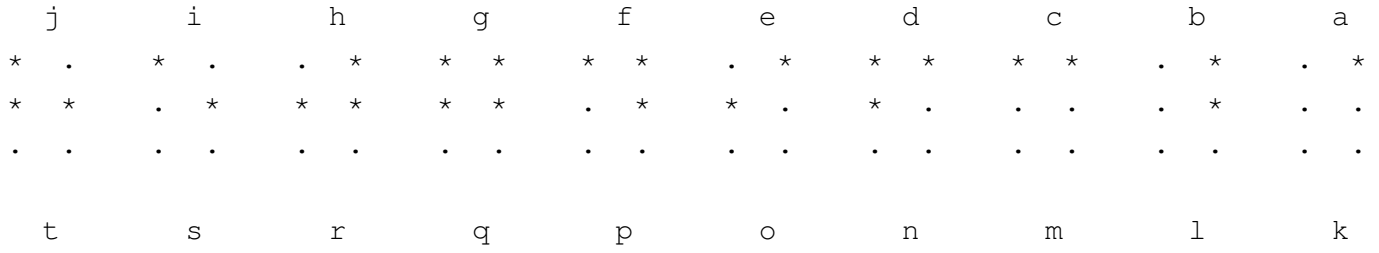
14.17.2 Braille a otto punti

La codifica a otto punti consente di utilizzare una sola cella per indicare una lettera maiuscola o un numero. Ciò diventa particolarmente importante in un terminale braille, in cui è bene che si possa mantenere una corrispondenza tra celle e caratteri.

14.17.3 Tabelle di conversione

L'abbinamento del braille all'informatica, impone l'introduzione di tabelle di conversione dalla codifica dei caratteri al braille stesso. In pratica, in base alla nazionalità, si utilizza una codifica particolare; oltre a questo, deve essere definito un abbinamento tra codice binario (ASCII, ISO 8859-***n***, ecc.) e rappresentazione braille. Quando si utilizza un terminale braille o un sistema di stampa braille, occorre stabilire anche questo tipo di conversione.

Figura 14.134. La codifica braille per la scrittura da destra verso sinistra, secondo l'ordine classico.



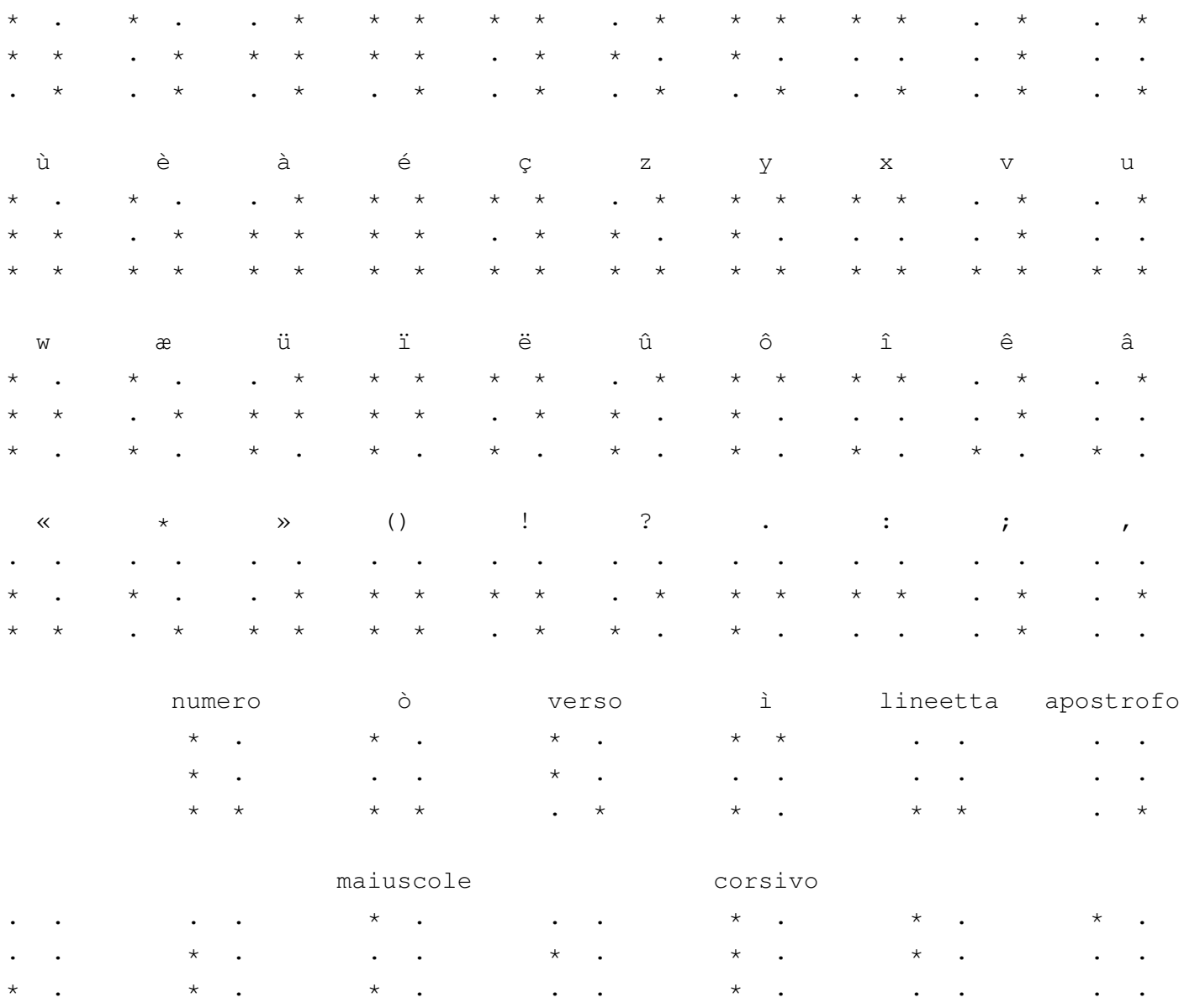
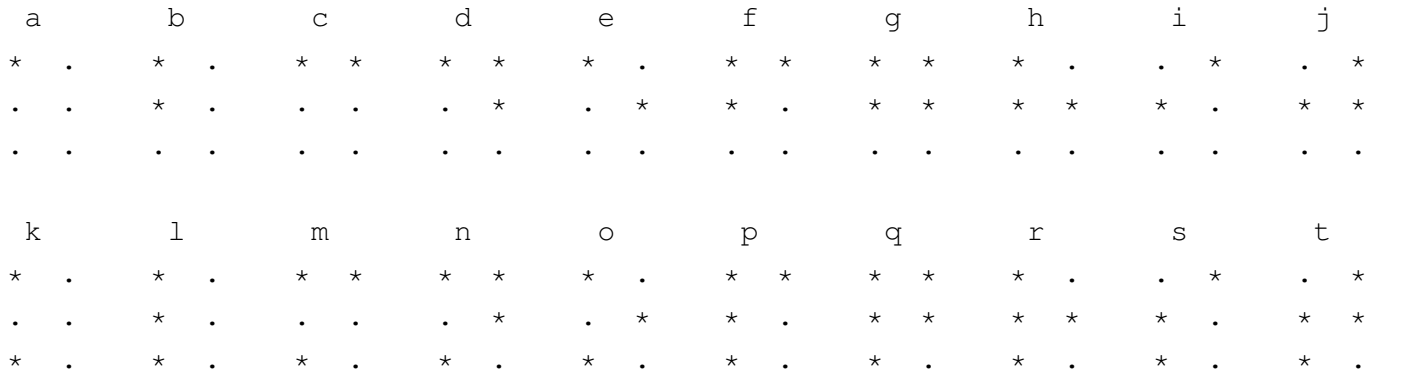
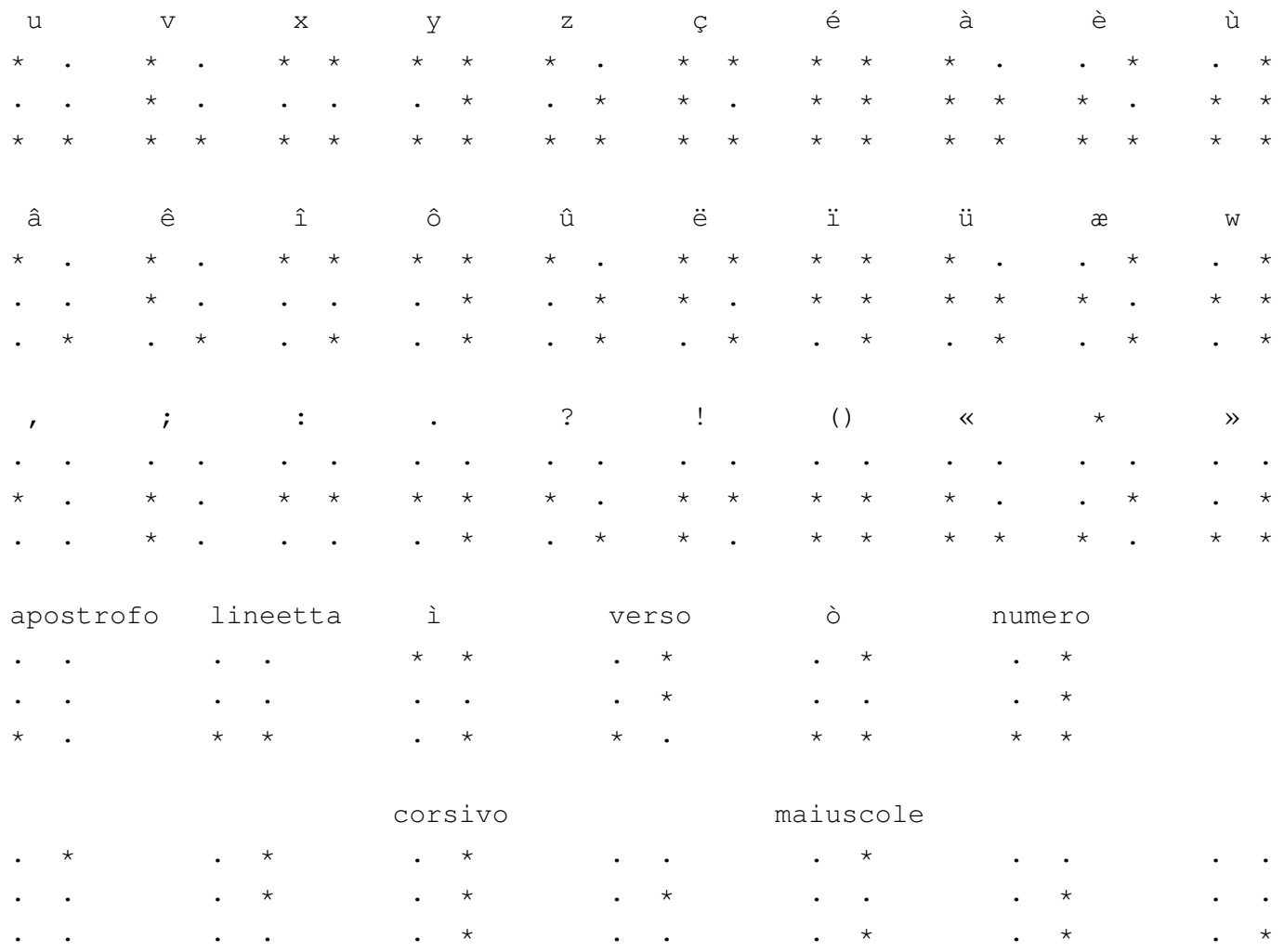


Figura 14.135. La codifica braille per la lettura da sinistra verso destra, secondo l'ordine classico.





14.17.4 Sistemi di interazione per non vedenti

«

I sistemi standard di interazione per non vedenti con un elaboratore sono di due tipi, ma si integrano convenientemente assieme: il terminale braille e la lettura automatica di ciò che appare sullo schermo. Il terminale braille è un'unità che si collega generalmente a una porta seriale o parallela e richiede la presenza di un programma in grado di leggere ciò che appare sullo schermo normale; il sistema di lettura automatica richiede naturalmente un programma simile.

Un sistema di lettura automatico è apparentemente un problema semplice da risolvere; in pratica le cose non sono così, perché deve anche essere in grado di individuare la struttura del testo (si pensi a

un'organizzazione in più colonne). Il sistema che sembra essere più efficace per la lettura sonora del testo è Emacspeack, che comunque qui non viene descritto. Si vedano eventualmente i riferimenti bibliografici alla fine del capitolo.

14.17.5 Lettura di ciò che appare sullo schermo

Qualunque sistema di lettura per non vedenti (braille o audio), deve avere un modo per accedere a ciò che appare sullo schermo. Esistono evidentemente due modi: ciò che appare sullo schermo è controllato direttamente dal programma di lettura, oppure questo programma deve interagire con il sistema operativo per ottenere una copia dello schermo.

Un programma che controlla ciò che si deve leggere potrebbe essere una shell, oppure una sorta di navigatore ipertestuale, che eventualmente incorpori anche delle funzionalità per la creazione e la modifica dei documenti (dovrebbe essere questo il caso di Emacspeack).

Nel caso di GNU/Linux si accede a una copia di ciò che appare sullo schermo della console attiva attraverso il file di dispositivo '`/dev/vcsa`' o '`/dev/vcsa0`'. In particolare questi file di dispositivo consentono di ottenere informazioni anche sugli attributi del testo, cosa che permette di localizzare il cursore, per esempio; esistono infatti anche i file di dispositivo '`/dev/vcs*`' che consentono di ottenere una pura copia testuale delle console virtuali.

Questi file di dispositivo sono normalmente inaccessibili a utenti diversi dall'amministratore; pertanto, per consentire una lettura a programmi con privilegi limitati, occorre intervenire nei permessi. Sup-

ponendo di dover creare da zero il file di dispositivo '`/dev/vcsa0`', probabilmente si dovrebbe agire come segue:

```
# mknod /dev/vcsa0 c 7 128 [Invio]

# chown root:tty /dev/vcsa0 [Invio]

# chmod 660 /dev/vcsa0 [Invio]
```

In pratica, si arriva alla fine a dare i permessi di lettura e scrittura anche al gruppo, supponendo che il programma che deve utilizzarlo funzioni, in questo caso, con i privilegi del gruppo '**`tty`**' (SGID-'**`tty`**'). Ciò consente di mantenere un minimo di riservatezza, nonostante il problema contingente da risolvere.

Volendo verificare che il file di dispositivo '`/dev/vcsa0`' consente di accedere al contenuto dello schermo della console virtuale attuale, si può inviarne una copia in una console inattiva o comunque non utilizzata. Per esempio:

```
# cat /dev/vcsa0 > /dev/tty12 [Invio]
```

Manda una copia alla 12-esima console virtuale e si può verificare facilmente con la combinazione di tasti [*Alt F12*].

14.17.6 Terminali braille

«

Il terminale braille è solitamente un componente da collocare al di sotto della tastiera normale, attraverso il quale è possibile leggere una riga di testo, senza allontanare troppo le dita dalla tastiera stessa.

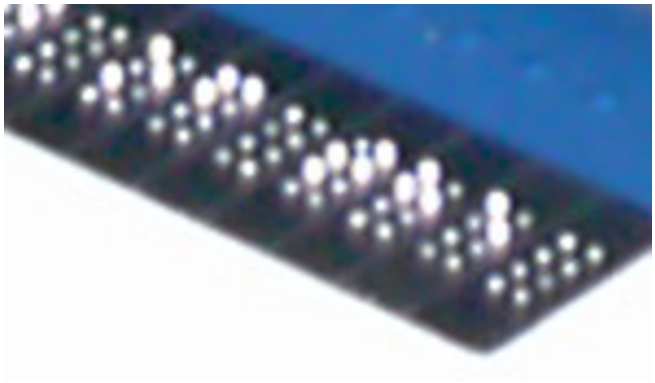
Vicino alla barra contenente le celle braille, devono essere disponibili alcuni tasti per il controllo del terminale stesso, per poter navigare sulla superficie di quello che appare sullo schermo. Infatti, di

solito si dispone di una sola riga di celle braille, ma anche se potessero essere disponibili più di una, non sarebbero mai quante le righe che si vedono su uno schermo normale; inoltre, anche la quantità di celle è limitata, per cui diventa necessario poter spostare la «visualizzazione» braille anche in orizzontale.

Figura 14.136. Collocazione tipica di un terminale braille, sotto la tastiera o sotto un elaboratore portatile.

La figura 14.137 mostra in particolare un dettaglio delle celle di un terminale braille a otto punti.

Figura 14.137. Dettaglio delle celle di un terminale braille.



14.17.7 Brltty

Brltty[23] è un programma in grado di comunicare con un terminale braille, allo scopo di inviargli ciò che accade sullo schermo della console virtuale attiva di un sistema GNU/Linux.

Il funzionamento del programma è relativamente semplice: si tratta di un demone, '**`brltty`**', da avviare e fermare automaticamente attraverso la procedura di inizializzazione del sistema, il quale comunica con il terminale braille seguendo la configurazione relativa al terminale stesso e la configurazione della conversione braille.

In pratica, Brltty deve conoscere il tipo di terminale braille utilizzato e deve disporre di una tabella di conversione adatta per quanto riguarda la codifica usata e l'alfabeto braille dell'utilizzatore.

Brltty è accompagnato da dei file di libreria, per l'adattamento alle caratteristiche del terminale braille e per la conversione finale della codifica in braille. Questi file dovrebbero essere collocati nella directory '`/etc/brltty/`'. Il file di configurazione in cui occorre intervenire dovrebbe essere '`/etc/brltty.conf`' e il suo contenuto potrebbe apparire come nell'esempio seguente:

```
braille-device /dev/ttyS0
braille-driver ec
dot-translation text.us.tbl
```

Si comprende intuitivamente il significato delle direttive: il terminale viene connesso alla prima porta seriale, corrispondente al file di dispositivo '`/dev/ttyS0`'; il terminale braille viene gestito attraverso un protocollo definito dalla sigla '**`ec`**'; la trasformazione della codifica in braille avviene secondo il modello denominato '**`text.us.tbl`**', corrispondente in pratica a un file contenuto nella directory '`/etc/brltty/`', facente riferimento a un adattamento per la lingua inglese.

Tabella 14.139. Sigle attraverso le quali si abbinano le librerie di controllo del terminale braille.

Sigla	Terminale corrispondente
`al`	Alva (ABT3xx/Delphi)
`bl`	BrailleLite 18
`b4`	BrailleLite 40
`cb`	Tieman CombiBraille
`ec`	EcoBraille
`eu`	EuroBraille
`md`	MDV braille
`pm`	Papenmeier

Sigla	Terminale corrispondente
`ts`	TSI (PowerBraille/Navigator)
`va`	BAUM Vario

Brltty è anche in grado di gestire terminali parlanti. Eventualmente, si può usare la direttiva '**`speech-driver`**' per il controllo della sintesi vocale. Di solito, la funzionalità è disabilitata, con la direttiva seguente:

```
speech-driver no
```

Tabella 14.141. Sigle relative alla gestione della sintesi vocale.

Sigla	Sintesi vocale
`no`	No Speech
`al`	Alva (Delphi)
`bl`	BrailleLite
`cb`	Tieman CombiBraille
`fv`	Festival Text to Speech Package
`gs`	Generic Say (invia allo standard input di '`/usr/local/bin/say`')
`tv`	Televox Speech Interface

Come già spiegato in precedenza, anche Brltty deve poter accedere a una copia della console virtuale corrente. Ciò avviene precisamente attraverso il file di dispositivo '`/dev/vcsa0`', che in alcuni siste-

mi GNU/Linux potrebbe anche essere assente, perché equivalente a '`/dev/vcsa`'. Si riepilogano nuovamente i passi necessari a creare il file e ad attribuirgli i permessi necessari per il funzionamento corretto con Brltty.

```
# mknod /dev/vcsa0 c 7 128 [Invio]

# chown root:root /dev/vcsa0 [Invio]

# chmod 600 /dev/vcsa0 [Invio]
```

In questo caso, si presume che il demone '**`brltty`**' sia avviato attraverso la procedura di inizializzazione del sistema, in modo tale da funzionare con tutti i privilegi dell'utente '**`root`**'. Diversamente, occorre cambiare strategia per quanto riguarda i permessi e la proprietà del file di dispositivo.

14.18 Riferimenti

«

- Richard Polt, *The classic typewriter page*, *http://site.xavier.edu/polt/typewriters/*
- Darryl Rehr, *The QWERTY connection*, *http://home.earthlink.net/~dcrehr/*
- *1000 bit*, *http://www.1000bi00t.it/ad/bro/brochures.asp*
- *Computermuseum München: Olivetti*, *http://www.computermuseum-muenchen.de/computer/olivetti/*
- R. A. Nelson, *History of teletype development*, 1963, *http://www.thocp.net/hardware/history_of_teletype_development_.htm* , *http://www.rtty.com/history/nelson.htm*

- Frank da Cruz, *Columbia University computing history, a chronology of computing at Columbia University*, http://www.columbia.edu/cu/computinghistory/
- *IBM Radiotype*, http://www-03.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_3.html , http://www-03.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_4.html
- *The historical meaning of ASCII Control Characters*, http://www.rootr.net/ascii_control_meaning.html
- Robert McConnell, James Haynes, Richard Warren, *Understanding ASCII Codes*, http://www.nadcomm.com/ascii_code.htm
- Jukka Korpela, *Ascii control codes (control characters, C0 controls)*, http://www.cs.tut.fi/~jkorpela/chars/c0.html
- Tom Jennings, *History of character codes*, http://wps.com/projects/codes/index.html
- *International register of coded character sets to be used with escape sequences*, http://www.itscj.ipsj.or.jp/ISO-IR/overview.htm
- *C0 control characters set*, 1987, http://www.itscj.ipsj.or.jp/ISO-IR/140.pdf
- *ISO/IEC 9995:1994, Information technology -- Keyboard layouts for text and office systems*, http://www.iso.org
- *Pictogrammes ISO 9995-7*, http://pages.infinit.net/pm2/lexique4.htm
- *Unicode, Miscellaneous Technical, Range: 2300-23FF*, *Unicode, Control Pictures, Range: 2400-2426*, http://www.unicode.org/

charts/PDF/U2300.pdf , *http://www.unicode.org/charts/PDF/U2400.pdf*

- Alex Buell, *Framebuffer HOWTO*, *http://tldp.org/HOWTO/html_single/Framebuffer-HOWTO/*
- Davi S. Lauer, Greg Hankins, *Serial HOWTO*, *http://www.tldp.org/HOWTO/html_single/Serial-HOWTO/*
- *Dottless braille*, *http://www.dotlessbraille.org/*
- *A (slightly) different introduction to braille*, *http://www.dotlessbraille.org/AboutBraille.htm*
- *BRLTTY - official home*, *http://dave.mielke.cc/brltty/*
- *BRLSPEAK: a braille and speech mini-distribution of GNU/Linux*, *http://www.brlspeak.net/*
- *Blind + Linux = BLINUX*, *http://leb.net/blinux/*

[1] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[2] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[3] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[4] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[5] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[6] **GNU core utilities** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **Ncurses** software libero con licenza speciale FSF

[9] **Ncurses** software libero con licenza speciale FSF

[10] Per terminare l'utilizzo di '**mc**' si può utilizzare la sequenza [*Esc*][*0*] se non funziona il tasto [*F10*].

[11] **Ncurses** software libero con licenza speciale FSF

[12] **Linux console tools** GNU GPL

[13] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[14] **Linux console tools** GNU GPL

[15] **General purpose mouse interface utility** GNU GPL

[16] **Script** UCB BSD

[17] **Open** GNU GPL

[18] **Switchto** GNU GPL

[19] **Screen** GNU GPL

[20] **Pconsole** GNU GPL

[21] **Mingetty** GNU GPL

[22] **Agetty** non specifica alcuna condizione, tuttavia viene indicato come «distribuibile liberamente» e di norma viene distribuito con la licenza GNU-GPL

[23] **Brltty** GNU GPL