22.1 Gestione dei file di testo

Un file di testo è quello che può essere letto così com'è senza bisogno di interpretazioni particolari. Il modo con cui questo file viene definito chiarisce anche il tipo di contenuto che può avere: testo puro, senza altre informazioni. I file di testo sono costituiti da una sequenza di caratteri e simboli, separata convenzionalmente in righe di lunghezze diseguali, la quale così deve essere terminata attraverso un codice particolare: quello che qui viene definito ***codice di interruzione di riga***.

22.1.1 Codice di interruzione di riga

Il codice di interruzione di riga cambia a seconda del sistema operativo. Negli ambienti Unix si usa il codice $0A_{16}$ che viene chiamato *<LF>*, negli ambienti Dos e derivati si utilizza la sequenza $0D0A_{16}$ corrispondente a *<CR><LF>*, mentre negli ambienti Mac si usa il codice $0D_{16}$ corrispondente a *<CR>*.

Quando si vuole fare riferimento al codice di interruzione di riga in modo astratto, cioè senza restare legati a un'architettura particolare del sistema operativo, si parla spesso di *new-line*. La convenzione per cui il termine *new-line* dovrebbe rappresentare idealmente ciò che si utilizza per interrompere una riga, non viene rispettata sempre. Generalmente, chi lavora con i sistemi Unix ignora il problema e utilizza il termine *new-line* per identificare il carattere *<LF>*, il quale ha invece un nome preciso: *line feed*.

Utilizzando un sistema GNU, il problema derivato dall'ambiguità tra il carattere *<LF>* e il concetto di *new-line*, non si manifesta; tuttavia, leggendo i documenti che utilizzano questa espressione, occorre fare attenzione, o almeno è il caso di porsi il dubbio sul significato di ciò che si intende. Pertanto, in questo documento si utilizza la definizione «codice di interruzione di riga», in modo da non lasciare dubbi.

22.1.2 Spazi

Nei file di testo, gli spazi sono un concetto prettamente visivo: quando si visualizza (o si stampa) un file si notano delle zone in cui non appaiono caratteri di alcun tipo. I caratteri attraverso i quali si ottengono questi spazi sono normalmente: lo spazio vero e proprio (30_{16}), la tabulazione (09_{16}) e il codice di interruzione di riga. Il modo con cui si ottiene una spaziatura dipende dal contesto. Ciò che conta è sapere distinguere tra spazio in senso generale e ***carattere spazio*** che è invece un codice particolare. A volte si utilizza il termine *blank*, o ***spazio lineare***, per indicare uno spazio in senso generale, dove

di norma si intende fare riferimento indifferentemente a un numero imprecisato di caratteri spazio o tabulazione.

Il codice di interruzione di riga entra in gioco quando si ha a che fare con righe vuote. Una riga bianca, nel senso di *blank line*, può contenere spazi di vario genere (spazio e tabulazione) e poi deve essere conclusa da questo codice. Ma una riga vuota, *empty line*, contiene soltanto il codice di interruzione di riga.

22.2 Riemissione completa di un file di testo

Alcuni programmi per l'elaborazione di file di testo emettono lo stesso file fornito come input, eventualmente dopo un qualche tipo di elaborazione elementare. Il più semplice di questi programmi è '**cat**' che nella maggior parte dei casi viene utilizzato senza opzioni, per lo più con lo scopo di iniziare un condotto di programmi.

22.2.1 Utilizzo di «cat»

Il programma di servizio '**cat**'[1] emette, attraverso lo standard output, il contenuto dei file indicati come argomento, oppure quanto proviene dallo standard input, in pratica fa qualcosa di simile al comando '**TYPE**' del Dos:

```
cat [opzioni] file...
```

```
cat [opzioni] < file
```

Tabella 22.1. Alcune opzioni.

Opzione	Descrizione
`-b` `--number-nonblank`	Numera tutte le righe emesse, che non sono vuote, a partire da uno.
`-s` `--squeeze-blank`	Sostituisce le righe vuote multiple con una sola riga bianca.
`-v` `--show-nonprinting`	Sostituisce i caratteri non stampabili con una sigla che inizia con un accento circonflesso ('**^**') o con la sigla '**M-**' a seconda che si tratti di un codice con il primo bit a zero oppure a uno (si intende il bit più significativo). Sono esclusi i caratteri di tabulazione e i codici di interruzione di riga.
`-E` `--show-ends`	Visualizza la fine di ogni riga aggiungendo il simbolo '**$**'.
`-T` `--show-tabs`	Mostra esplicitamente il carattere di tabulazione (<*HT*>) utilizzando il simbolo '**^I**'.
`-A` `--show-all`	Mostra tutti i simboli non stampabili. È equivalente a '**-vET**'.

22.2.2 Utilizzo di «tac»

Il programma di servizio '**tac**'[2] emette attraverso lo standard output il contenuto dei file forniti come argomento, oppure quanto proveniente dallo standard input, ma invertendo l'ordine delle righe:

```
tac [opzioni] file...
```

```
tac [opzioni] < file
```

Queste righe possono essere divise in base a un codice di interruzione di riga diverso dal solito, specificandolo attraverso l'opzione '**-s**'. In pratica, '**tac**' è l'inverso di '**cat**'.

Opzione	Descrizione
`-s` ***separatore*** `--separator=`***separatore***	Permette di definire il codice di interruzione di riga da prendere in considerazione.

22.2.3 Utilizzo di «nl»

Il programma di servizio '**nl**'[3] (*number line*) emette attraverso lo standard output il contenuto dei file forniti come argomento, oppure quanto proveniente dallo standard input, con l'aggiunta dei numeri di riga per alcune o tutte le righe dell'input:

```
nl [opzioni] file...
```

```
nl [opzioni] < file
```

Il conteggio delle righe viene fatto unendo il contenuto di tutti i file forniti come argomento, come se si trattasse di un file unico, azzerandolo ogni volta che viene riconosciuto l'inizio di una pagina logica. Sotto questo punto di vista, una pagina logica è composta da tre parti: testa, corpo e piede.

In condizioni normali, salvo altre indicazioni passate attraverso le opzioni, si individua la testa, il corpo e il piede, perché preceduti da una riga che contiene esattamente una stringa speciale:

- `\:\:\:` prima della testa;
- `\:\:` prima del corpo;
- `\:` prima del piede.

Per mostrare semplicemente il funzionamento del programma, viene proposto un file di testo puro contenente esattamente le righe seguenti:

```
\:\:\:
intestazione 1
intestazione 2
intestazione 3
\:\:
corpo 1
corpo 2
corpo 3
corpo 4
corpo 5
\:
piede 1
piede 2
piede 3
\:\:\:
intestazione 4
intestazione 5
intestazione 6
\:\:
corpo 6
corpo 7
corpo 8
corpo 9
corpo 10
\:
piede 4
piede 5
piede 6
```

Supponendo che il file abbia il nome '`prova.txt`', ecco cosa si ottiene con il comando seguente:

```
$ nl prova.txt [Invio]

        intestazione 1
        intestazione 2
```

```
    intestazione 3

1   corpo 1
2   corpo 2
3   corpo 3
4   corpo 4
5   corpo 5

    piede 1
    piede 2
    piede 3

    intestazione 4
    intestazione 5
    intestazione 6

1   corpo 6
2   corpo 7
3   corpo 8
4   corpo 9
5   corpo 10

    piede 4
    piede 5
    piede 6
```

Si osservi in particolare che la prima riga è vuota, perché nel file di origine si trova in quella posizione la stringa di riconoscimento dell'inizio della testa.

Si veda eventualmente il documento *info nl* oppure la pagina di manuale *nl(1)*.

22.2.4 Utilizzo di «od»

Il programma di servizio '**od**'[4] converte i file forniti come input in ottale o in altri formati.

```
od [opzioni] file…
```

```
od [opzioni] < file
```

Si veda eventualmente il documento *info od* oppure la pagina di manuale *od(1)*.

22.2.5 Utilizzo di «rev»

Il programma di servizio '**rev**'[5] (*revert*) emette attraverso lo standard output il file fornito come argomento, invertendo l'ordine dei caratteri di ogni riga:

```
rev [file…]
```

22.2.6 Utilizzo di «dog»

Il programma di servizio '**dog**'[6] è una rivisitazione di '**cat**', a cui aggiunge in particolare la possibilità di accedere a URI esterni:

```
dog [opzioni] [- | file | uri]…
```

Vengono mostrati alcuni esempi.

- `$ dog prova.txt` [*Invio*]

  Legge il file '`prova.txt`' ed emette il suo contenuto attraverso lo standard output.

- `$ dog http://a2.brot.dg/prova.html` [*Invio*]

  Preleva una copia di *http://a2.brot.dg/prova.html* ed emette l'esito attraverso lo standard output. La parte iniziale di ciò che si ottiene potrebbe apparire così:


```
HTTP/1.0 200 OK
Date: Tue, 21 Mar 2006 16:46:00 GMT
Server: Apache/2.0.54 (Debian GNU/Linux) PHP/4.3.10-16
Last-Modified: Wed, 15 Mar 2006 07:36:51 GMT
ETag: "4d9a3f-3222-a39aeac0"
Accept-Ranges: bytes
Content-Length: 12834
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
…
```

- `$ dog --no-header http://a2.brot.dg/prova.html > prova.html` [*Invio*]

  Preleva una copia di *http://a2.brot.dg/prova.html* e la salva nel file '`prova.html`'. L'uso dell'opzione '**`--no-header`**' serve a ottenere il file originale, senza l'aggiunta dell'intestazione che fa parte del protocollo HTTP.

Si veda la pagina di manuale *dog(1)*.

22.3 Rimpaginazione di un file di testo

Alcuni programmi si occupano di modificare l'impaginazione del testo, cambiandone la larghezza o aggiungendo delle intestazioni.

22.3.1 Utilizzo di «fmt»

Il programma di servizio '**fmt**'[7] (*format*) impagina il testo contenuto nei file forniti come argomento, eliminando e aggiungendo codici di interruzione di riga, in modo che il testo risulti al massimo di una data larghezza:

```
fmt [opzioni] [file…]
```

Il risultato viene emesso attraverso lo standard output. Se non si specifica l'ampiezza massima della riga, questa si intende essere di 75 caratteri.

A titolo di esempio, si suppone che il file '`prova.txt`' contenga esattamente il testo seguente:

```
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla

bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
```

Con '**fmt**' si ottiene la rimpaginazione dei due blocchi:

```
$ fmt prova.txt [Invio]

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla
```

Opzione	Descrizione
`-w` *ampiezza* `--width=`*ampiezza*	Permette di specificare l'ampiezza massima del testo.

Si veda eventualmente il documento *info fmt* oppure la pagina di manuale *fmt(1)*.

22.3.2 Utilizzo di «pr»

Il programma di servizio '**pr**'[8] (*print*) emette attraverso lo standard output il contenuto dei file forniti come argomento, formattati opportunamente per la stampa:

```
pr [opzioni] [file...]
```

Se non viene indicato diversamente attraverso le opzioni:

- viene aggiunta un'intestazione di cinque righe, dove la terza di queste contiene l'informazione della data e dell'ora di stampa e anche del numero di pagina;
- viene aggiunto un piede di pagina di quattro righe bianche;
- il numero di righe per pagina è di 66;
- il carattere <*FF*> che fosse contenuto eventualmente nei file di input, genera un salto pagina.

Nel caso si utilizzino più colonne:

- le colonne hanno la stessa larghezza;
- sono separate da una stringa opzionale (il cui valore predefinito è uno spazio);
- le righe vengono troncate al raggiungimento della larghezza massima della colonna.

Tabella 22.12. Alcune opzioni.

Opzione	Descrizione
+pagina_iniziale [*:pagina_finale*]	Seleziona l'intervallo di pagine indicato. Se manca la pagina finale, si intende che sia l'ultima.
-numero_colonne	Produce un risultato suddiviso nel numero di colonne indicato. L'ampiezza delle colonne viene calcolata automaticamente.
`-c`	Stampa i caratteri di controllo utilizzando la notazione: '**^A**', '**^B**',... I simboli comunque non stampabili vengono rappresentati in ottale: '***ooo***'.
`-d`	Raddoppia la spaziatura tra le righe (spaziatura doppia).
`-l` *altezza_pagina*	Definisce l'altezza della pagina espressa in righe. Di solito la pagina è alta 66 righe.
`-o` *margine_sinistro*	Definisce il margine sinistro espresso in caratteri.

Si veda eventualmente il documento *info pr* oppure la pagina di manuale *pr(1)*.

22.3.3 Utilizzo di «fold»

Il programma di servizio '**fold**'[9] impagina il testo contenuto nei file forniti come argomento, suddividendo le righe troppo lunghe inserendo un codice di interruzione di riga al raggiungimento della colonna 80 (se non viene specificato diversamente attraverso le opzioni):

```
fold [opzioni] [file...]
```

Il conteggio viene fatto tenendo conto dei caratteri di tabulazione come se fossero espansi, dei codici di *backspace* (<*BS*>) che diminuiscono il conteggio e dei caratteri di ritorno a carrello (<*CR*>) che azzerano il conteggio.

Tabella 22.13. Alcune opzioni.

Opzione	Descrizione
`-b` `--bytes`	Conta i caratteri <*HT*> (tabulazione), <*BS*> e <*CR*> come gli altri.
`-s` `--spaces`	Cerca di interrompere le righe subito dopo uno spazio vuoto, in modo da evitare di spezzare delle parole.

Opzione	Descrizione
`-w` *larghezza* `--width=`*larghezza*	Definisce la larghezza massima della colonna di testo finale.

22.3.4 Utilizzo di «column»

Il programma di servizio '**column**'[10] trasforma l'input rappresentato dai file indicati come argomento in modo da ottenere più colonne. Il risultato viene emesso attraverso lo standard output:

```
column [opzioni] [file...]
```

Tabella 22.14. Alcune opzioni.

Opzione	Descrizione
`-c` *colonne*	Indica la dimensione orizzontale in caratteri dell'output che si vuole ottenere.
`-x`	Di norma, ogni colonna viene riempita completamente prima di passare alla successiva. Con questa opzione, i dati all'interno delle colonne vengono disposti in sequenza orizzontale.

22.3.5 Utilizzo di «colrm»

Il programma di servizio '**colrm**'[11] elimina un intervallo di colonne da un file di testo:

```
colrm [colonna_iniziale] [colonna_finale]
```

Le colonne sono espresse in caratteri. L'input è ottenuto dallo standard input e l'output viene emesso attraverso lo standard output. Se vengono omessi gli attributi, non viene eliminato alcunché. Se manca l'indicazione della colonna finale, l'eliminazione avviene a partire dalla colonna iniziale indicata fino alla fine della riga.

Segue la descrizione di alcuni esempi.

- `$ colrm 1 10 < documento.txt > tagliato.txt` [*Invio*]

  Vengono eliminati i primi 10 caratteri di ogni riga del file '`documento.txt`' e il risultato viene emesso in '`tagliato.txt`'.

- `$ colrm 81 < documento > normale.txt` [*Invio*]

  Le righe che superano gli 80 caratteri vengono tagliate.

22.4 Composizione

Un tipo di file di testo è quello che contiene codici speciali di spostamento allo scopo di ottenere un aspetto particolare quando questo viene stampato. Il codice più usato in questo senso è <*BS*> (*backspace*) che normalmente viene interpretato come arretramento di una posizione. Gli effetti che si possono ottenere in questo modo sono il sottolineato e il neretto. Per esempio, per ottenere la parola «CIAO» in neretto, si utilizza una sequenza del tipo:

'**C**<*BS*>**CI**<*BS*>**IA**<*BS*>**AO**<*BS*>**O**'.

22.4.1 Utilizzo di «col» e di «colcrt»

I programmi '**col**' e '**colcrt**' interpretano nei file di testo dei codici di spostamento particolari, oltre a quelli comuni della codifica ASCII:[12]

Codice	Descrizione
<*ESC*>7	Arretramento di una riga.
<*ESC*>8	Arretramento di mezza riga.
<*ESC*>9	Avanzamento di una riga.
<*VT*>	Arretramento di una riga (come <*ESC*>7).

Il programma di servizio '**col**'[13] emette attraverso lo standard out-

put una trasformazione dello standard input, in modo che questo contenga solo codici per l'avanzamento di riga in avanti.

Tabella 22.16. Alcune opzioni.

Opzione	Descrizione
`-b`	Vengono eliminati anche i caratteri di *backspace* (<*BS*>).
`-f`	I caratteri di mezzo *line feed* (avanzamento di mezza riga) vengono permessi e quindi non sono eliminati.
`-x`	I caratteri di tabulazione vengono sostituiti con spazi.

L'esempio seguente trasforma il risultato di *ls(1)* in un file di testo normale, memorizzato in '`/tmp/ls1.txt`':

```
$ man 1 ls | col -bx > /tmp/ls1.txt [Invio]
```

Il programma di servizio '**colcrt**'[14] emette attraverso lo standard output una trasformazione dei file indicati come argomento, in modo da permettere la visualizzazione o la stampa di testi contenenti codici di spostamento di mezza riga:

```
colcrt [opzioni] [file...]
```

In pratica, i caratteri che dovrebbero apparire stampati in una mezza riga successiva, vengono spostati nella riga (intera) successiva. Il suo funzionamento è simile a quello del programma '**col**', ma è orientato particolarmente all'emissione di un output adatto allo schermo di un terminale.

Tabella 22.17. Alcune opzioni.

Opzione	Descrizione
`-`	Vengono eliminate tutte le sottolineature.
`-2`	Stampa tutte le mezze righe, raddoppiando in pratica l'output.

22.4.2 Utilizzo di «ul»

Il programma di servizio '**ul**',[15] il cui nome sta per *underline*, permette di trasformare un file contenente codici di arretramento, usati per ottenere evidenziamenti e sottolineature, in modo da poterlo visualizzare correttamente in base al tipo di terminale che risulta dalla configurazione (la variabile di ambiente ***TERM***):

```
ul [opzioni] [file...]
```

Per comprendere il senso di questo programma, si può ridirigere su un file il risultato della lettura di una pagina di manuale:

```
$ man 1 ul > /tmp/ul.txt [Invio]
```

Se si osserva il file '`/tmp/ul.txt`' ottenuto, si può vedere che contiene dei codici di arretramento (<*BS*>), con lo scopo di ottenere un carattere di spessore maggiore (più nero, o più illuminato, a seconda del modo con cui il terminale funziona normalmente):

```
UL(1)                    BSD General Commands Manual                   UL(1)

N<BS>NA<BS>AM<BS>ME<BS>E
     u<BS>ul<BS>l - do underlining

S<BS>SY<BS>YN<BS>NO<BS>OP<BS>PS<BS>SI<BS>IS<BS>S
     u<BS>ul<BS>l [-<BS>-i<BS>i] [-<BS>-t<BS>t _<BS>t_<BS>e_<BS>r_<BS>m_<BS>
↪<BS>i_<BS>n_<BS>a_<BS>l] [_<BS>n_<BS>a_<BS>m_<BS>e _<BS>._<BS>._<BS>.]

D<BS>DE<BS>ES<BS>SC<BS>CR<BS>RI<BS>IP<BS>PT<BS>TI<BS>IO<BS>ON<BS>N
     The u<BS>ul<BS>l utility reads the named files (or standard input
     if none are given) and translates occurrences of underscores to
     the sequence which indicates underlining for the terminal in use,
     …
```

Se si invia questo file, così come si vede, allo schermo, non si ottiene alcun evidenziamento:

```
$ cat /tmp/ul.txt [Invio]
```

Se invece si utilizza '**ul**', lo si vede come dovrebbe (purché il terminale sia in grado di farlo):


```
$ ul /tmp/ul.txt [Invio]
```

Eventualmente si può consultare la pagina di manuale *ul(1)*.

22.5 Estrazione parziale e statistiche

Alcuni programmi si occupano di estrarre dall'input solo alcune porzioni, o solo delle informazioni riepilogative, o ancora di suddividere l'input in parti più piccole.

22.5.1 Utilizzo di «head»

Il programma di servizio '**head**'[16] emette attraverso lo standard output la prima parte (le prime 10 righe se non viene specificato diversamente con le opzioni) dei file forniti come argomento:

```
head [opzioni] [file...]
```

Tabella 22.19. Alcune opzioni.

Opzione	Descrizione
`-c` ***dimensione*** `--bytes=`***dimensione***	Emette la quantità di byte indicata dalla dimensione.
`-n` ***dimensione*** `--lines=`***dimensione***	Emette la quantità di righe indicata dalla dimensione.

22.5.2 Utilizzo di «tail»

Il programma di servizio '**tail**'[17] emette attraverso lo standard output la parte finale (le ultime 10 righe se non viene specificato diversamente con le opzioni) dei file forniti come argomento:

```
tail [opzioni] [file...]
```

Tabella 22.20. Alcune opzioni.

Opzione	Descrizione
`-c` ***n*** `--bytes=`***n***	Se l'argomento è privo di segno, emette gli ultimi ***n*** byte del file; se invece l'argomento è preceduto da un segno «+», si intende ottenere la porzione di file che inizia dal byte ***n***-esimo (il primo corrispondente a +1) fino alla fine.
`-n` ***n*** `--lines=`***n***	Se l'argomento è privo di segno, emette le ultime ***n*** righe del file; se invece l'argomento è preceduto da un segno «+», si intende ottenere la porzione di file che inizia dalla riga ***n***-esima (la prima corrispondente a +1) fino alla fine.
`-f` `--follow`	Cerca di continuare la lettura del file, assumendo che questo debba allungarsi per opera di un altro processo (come avviene nel caso dei file delle registrazioni).

L'esempio seguente legge la parte finale del file '`/var/log/messages`' e continua a farlo in attesa di aggiunte al file, inviando quanto ottenuto al dispositivo '`/dev/tty10`'. Il processo viene messo opportunamente sullo sfondo in modo da liberare il terminale.

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

22.5.3 Utilizzo di «split»

Il programma di servizio '**split**'[18] suddivide il contenuto del file fornito come argomento in porzioni ordinate, di una lunghezza massima definita (di solito 1000 righe):

```
split [opzioni] [file [prefisso]]
```

I file che vengono generati hanno il prefisso indicato nell'argomento (oppure '**x**' se non viene specificato), seguito da una coppia di lettere che cambiano secondo una sequenza ordinata: '**aa**', '**ab**', '**ac**',...

Pertanto, i nomi ottenuti possono essere ordinati nello stesso modo con cui il file originale è stato suddiviso.

Tabella 22.21. Alcune opzioni.

Opzione	Descrizione
`-l` ***righe*** `--lines=`***righe***	Definisce il numero di righe dei file di destinazione.
`-b` ***dimensione*** `--bytes=`***dimensione***	Definisce la dimensione in byte dei file di destinazione.
`-C` ***dimensione*** `--line-bytes=`***dimensione***	Definisce la dimensione massima in byte dei file di destinazione, intendendo però che questi file contengano righe intere. Si possono usare gli stessi moltiplicatori utilizzabili nell'opzione '**`-b`**'.

22.5.4 Utilizzo di «csplit»

Il programma di servizio '**`csplit`**'[19] serve a suddividere un file in più parti, secondo uno o più modelli. Ogni modello indicato alla fine della riga di comando, specifica l'inizio di una nuova porzione di file che si vuole ottenere:

`csplit` [***opzioni***] ***file modello***...

Il modello in questione può essere rappresentato da un numero che esprime la quantità di righe da contare, per determinare l'inizio della porzione successiva, oppure da un'espressione regolare che, a seconda della delimitazione, può anche servire a eliminare una porzione di file.

Salvo indicazione diversa, '**`csplit`**' genera una serie di file secondo il modello '`xx`***`nn`***'; in pratica, si tratta di file il cui nome inizia con due «x» e termina con un numero di due cifre. In particolare, con l'opzione '**`-f`**' è possibile stabilire un prefisso diverso da queste due «x».

Come accennato, ogni modello può essere rappresentato da un numero, o da un'espressione regolare, ma in più, può essere aggiunto un simbolo speciale che ne rappresenta la ripetizione, per permettere l'indicazione dello stesso modello per più porzioni del file:

modello [{***n_ripetizioni***|`*`}]

Come si vede dallo schema sintattico, l'indicazione eventuale delle ripetizioni è racchiuso tra parentesi graffe, che quindi fanno parte del comando. Quando si vogliono indicare tali parentesi graffe, è probabile che si debbano proteggere dall'interpretazione della shell. Vengono descritte di seguito le varie forme dei modelli che possono essere indicati, assieme a una spiegazione sull'uso del simbolo di ripetizione.

Tenendo conto che dovrebbe trattarsi di un programma di servizio GNU, le espressioni regolari vanno espresse secondo le convenzioni GNU. Per la precisione, vengono utilizzate le espressioni regolari estese (ERE). A questo proposito è conveniente leggere le sezioni 23.1 e 23.2.

- ***n_righe***

  Un numero puro e semplice, indica di concludere la porzione di file in corso all'***n***-esima riga.

- ***/espressione_regolare/*** [***scostamento***]

  Un'espressione regolare racchiusa tra due barre oblique normali, serve a indicare la riga di inizio che deve appartenere alla prossima porzione di file. In pratica, la riga per la quale si avvera la corrispondenza, è la prima della porzione di file successiva. Se dopo la seconda barra obliqua si indica un numero preceduto da un segno '**`+`**' o da un segno '**`-`**', significa che si vuole indicare uno scostamento da quel punto, espresso in caratteri: uno scostamento positivo sposta l'inizio dopo ***n*** caratteri dall'inizio della riga trovata; uno scostamento negativo, sposta l'inizio all'indietro di ***n*** caratteri.

- ***%espressione_regolare%*** [***scostamento***]

  Un'espressione regolare racchiusa tra due simboli di percentuale, fa in modo che la porzione successiva, il cui inizio viene identificato attraverso l'espressione regolare stessa, non viene inserito in un file. In pratica, questa parte viene ignorata semplicemente. L'indicazione eventuale dello scostamento si comporta nello stesso modo già visto per il caso precedente.

- ***{n}***

  `{*}`

  Il simbolo di ripetizione fa in modo che il modello precedente venga ripetuto ***n*** volte, oppure, se si usa un asterisco, tante volte quante ne servono a completare la scansione del file in ingresso.

Tabella 22.22. Alcune opzioni.

Opzione	Descrizione
`-f` ***prefisso_nomi_file*** `--prefix=`***prefisso_nomi_file***	Questa opzione permette di specificare l'inizio dei nomi dei file che vengono generati. Senza usare questa opzione, i file iniziano per '`xx*`'.
`-n` ***n_cifre_numeriche*** `--digits=`***n_cifre_numeriche***	In condizioni normali, il nome dei file che vengono generati, termina con due sole cifre numeriche, cosa che consente la creazione di 100 file differenti (da 00 a 99). Con questa opzione è possibile modificare il numero di cifre numeriche di questi file.

`-s`
`-q`
`--silent`
`--quiet`
	Se non viene usata questa opzione, '**`csplit`**' emette attraverso lo standard output una serie di numeri, corrispondenti alla dimensione in byte dei file che vengono generati.

Segue la descrizione di alcuni esempi.

- `$` **`csplit elenco 10`** [*Invio*]

  Legge il file '`elenco`' generando il file '`xx00`' con le prime 10 righe di questo, quindi il file '`xx01`' con le righe restanti.

- `$` **`csplit elenco 10 \{0\}`** [*Invio*]

  Questo esempio serve a mostrare l'uso del simbolo di ripetizione: in questo caso, '**`{0}`**', con le parentesi graffe debitamente protette, non ha effetto e il risultato è lo stesso dell'esempio precedente.

- `$` **`csplit elenco 10 \{1\}`** [*Invio*]

  Legge il file '`elenco`' generando il file '`xx00`' con le prime 10 righe di questo, il file '`xx01`' con le 10 righe successive e il file '`xx02`' con le righe restanti.

- `$` **`csplit -f prova elenco 10 \{1\}`** [*Invio*]

  Come nell'esempio precedente, con la differenza che i file generati iniziano per '`prova*`'.

- `$` **`csplit -f prova elenco '/Pagina [0-9]*/' \{\*\}`** [*Invio*]

  Suddivide il file '`prova`', in modo che ogni porzione inizi con una riga corrispondente all'espressione regolare '**`Pagina [0-9]*`**'. La ricerca della corrispondenza dell'espressione regolare avviene per tutte le righe disponibili, in base alla presenza del simbolo di ripetizione indefinita: '**`{*}`**'. I file generati hanno tutti la radice '`prova`'.

22.5.5 Utilizzo di «wc»

Il programma di servizio '**wc**'[20] (*word count*) emette attraverso lo standard output la statistica sul conteggio dei codici di interruzione di riga (in pratica il numero delle righe), delle parole e dei byte:

```
wc [opzioni] [file...]
```

Se attraverso le opzioni vengono specificati uno o due tipi di conteggio, quelli che non sono indicati espressamente non vengono emessi.

Tabella 22.23. Alcune opzioni.

Opzione	Descrizione
`-c` `--bytes`	Emette la dimensione in byte.
`-w` `--words`	Emette il totale delle parole.
`-l` `--lines`	Emette il numero di righe (precisamente il numero dei codici di interruzione di riga).

22.6 Ordinamento di un file di testo

Alcuni programmi si occupano di riordinare file o di utilizzare file ordinati. L'ordinamento, la fusione, l'eliminazione degli elementi doppi e la ricerca binaria, rientrano in questo concetto.

22.6.1 Utilizzo di «sort»

Il programma di servizio '**sort**'[21] permette di ordinare o fondere insieme (*merge*) il contenuto dei file:

```
sort [opzioni] [file...]
```

Sono disponibili tre modalità di funzionamento:

- ordinamento (è la modalità predefinita);
- controllo dell'ordinamento;
- fusione.

Il risultato dell'ordinamento o della fusione, viene emesso attraverso lo standard output se non si specifica diversamente attraverso le opzioni; se non si specificano file da elaborare, viene preso in considerazione lo standard input.

Tabella 22.24. Opzioni riferite alla modalità di funzionamento.

Opzione	Descrizione
`-c`	Controlla che i file indicati siano già ordinati; se non lo sono, viene emessa una segnalazione di errore e il programma termina restituendo il valore uno, corrispondente a *Falso*.
`-m`	Fonde insieme i file indicati che devono essere già stati ordinati in precedenza. Nel caso non lo siano, si può sempre usare la modalità di ordinamento normale. L'utilizzo di questa opzione fa risparmiare tempo quando la situazione lo consente.

Tabella 22.25. Opzioni riferite al tipo di ordinamento.

Opzione	Descrizione
`-b`	Ignora gli spazi bianchi iniziali.
`-d`	Durante l'ordinamento ignora tutti i caratteri che non siano lettere, numeri e spazi.

Opzione	Descrizione
`-f`	Non distingue tra maiuscole e minuscole.
`-i`	Ignora i caratteri speciali al di fuori dell'ASCII puro (da 30_{16} a $7E_{16}$ compresi).
`-n`	Esegue una comparazione, o un ordinamento, di tipo numerico, tenendo conto anche del segno meno e del punto decimale.
`-r`	Inverte l'ordine della comparazione o dell'ordinamento.

Tabella 22.26. Altre opzioni.

Opzione	Descrizione
`-o` *file_generato*	Invece di utilizzare lo standard output per emettere il risultato dell'ordinamento o della fusione, utilizza il file indicato come argomento di questa opzione.
`-t` *carattere_separatore*	Permette di definire il carattere usato come separatore tra i campi che compongono le varie righe (record).
`-u`	Il risultato dell'utilizzo di questa opzione dipende dalla modalità di funzionamento di '**sort**'. Se è attiva la modalità di ordinamento o di fusione, fa sì che, in caso di chiavi duplicate, venga emesso solo il primo di questi record. Se è attiva la modalità di controllo, fa sì che venga segnalato un errore in presenza di chiavi duplicate.
`-k` *campo_iniziale* [*, campo_finale*]	Specifica la chiave di ordinamento o di fusione indicando il campo iniziale ed eventualmente quello finale. La suddivisione in campi può essere precisata con l'opzione '**-t**'.

Segue la descrizione di alcuni esempi.

- `$ sort /etc/passwd` [*Invio*]

  Riordina il file '`/etc/passwd`' a partire dalla prima colonna; in pratica, dal momento che si tratta del file che contiene le informazioni sulle utenze del sistema, lo riordina in base al nominativo di ognuna di queste (il primo campo).

- `$ sort -t : -k 1,1 /etc/passwd` [*Invio*]

  L'effetto di questo comando è praticamente identico a quello precedente, con la differenza che viene dichiarato esplicitamente l'intervento nel primo campo del file (come è noto, il file '`/etc/passwd`' è diviso in campi separati dai due punti verticali), dopo averne specificato il carattere separatore.

- `$ sort -n -t : -k 3,3 /etc/passwd` [*Invio*]

  Riordina il file già descritto, usando il terzo campo come chiave. In particolare, utilizza un ordinamento di tipo numerico, dal momento che il campo in questione rappresenta il numero UID di ogni utenza.

- `$ sort -m primo secondo > terzo` [*Invio*]

  Fonde assieme i due file '`primo`' e '`secondo`', già ordinati in precedenza, generando il file '`terzo`'.

- `$ sort -u -m primo secondo > terzo` [*Invio*]

  Come nell'esempio precedente, ma in questo caso, il file che viene generato non contiene righe doppie.

Si veda eventualmente il documento *info sort* oppure la pagina di manuale *sort(1)*.

22.6.2 Utilizzo di «uniq»

Il programma di servizio '**uniq**'[22] filtra il contenuto dei file ed emette solo le righe uniche. Il file fornito come input deve essere ordinato:

```
uniq [opzioni] [file_in_ingresso [file_in_uscita]]
```

Si veda eventualmente il documento *info uniq* oppure la pagina di manuale *uniq(1)*.

22.6.3 Utilizzo di «comm»

Il programma di servizio '**comm**'[23] confronta due file ordinati ed emette attraverso lo standard output l'indicazione delle righe uniche nel primo e nel secondo file, oltre alle righe che i due file hanno in comune:

```
comm [opzioni] [file1] file2
```

Se non vengono specificate delle opzioni, viene emesso un risultato su tre colonne: la prima contiene le righe uniche del primo file, la seconda le righe uniche del secondo file, la terza le righe in comune.

Tabella 22.27. Alcune opzioni.

Opzione	Descrizione
`-1`	Sopprime la prima colonna.
`-2`	Sopprime la seconda colonna.
`-3`	Sopprime la terza colonna.

22.7 Campi

Alcuni programmi si occupano di elaborare porzioni di file a livello delle righe (o dei record). Quando le righe di un file contengono informazioni strutturate in qualche modo, gli elementi di queste sono chiamati campi, inoltre, al posto del termine «riga» si preferisce utilizzare la parola record che esprime più precisamente il ruolo di questa: contenere una registrazione.

I campi di un record possono avere una dimensione fissa, oppure variabile. Nel primo caso anche i record hanno una dimensione fissa e la suddivisione in campi avviene in base alla posizione; nel secondo caso i record hanno una dimensione variabile e i campi vengono riconosciuti in base a un separatore che di solito deve essere definito.

22.7.1 Utilizzo di «cut»

Il programma di servizio '**cut**'[24] emette attraverso lo standard output porzioni del contenuto di ogni riga dei file indicati come argomento, o dello standard input in loro mancanza. Il modo con cui ciò avviene dipende dagli argomenti, attraverso i quali possono essere definite delle liste di valori o di intervalli. Il primo elemento corrisponde al numero uno.

```
cut [opzioni] [file...]
```

Tabella 22.28. Alcune opzioni.

Opzione	Descrizione
`-b` ***lista_di_byte*** `--bytes=`***lista_di_byte***	Definisce gli intervalli da estrarre espressi in byte.
`-c` ***lista_di_caratteri*** `--characters=`***lista_di_caratteri***	Definisce gli intervalli da estrarre espressi in caratteri.
`-f` ***lista_di_campi*** `--fields=`***lista_di_campi***	Definisce gli intervalli da estrarre espressi in campi. I campi sono distinti in base a un certo carattere usato come delimitatore. Quello predefinito è il carattere di tabulazione.
`-d` ***delimitatore*** `--delimiter=`***delimitatore***	Definisce un delimitatore alternativo al carattere di tabulazione.

- `$ cut -c 1-10 pippo` [*Invio*]

  Emette i primi 10 caratteri di ogni riga del file '`pippo`'.

- `$ cut -c 1-10,21 pippo` [*Invio*]

  Emette per ogni riga del file '`pippo`' solo i primi 10 caratteri seguiti dal 21-esimo carattere.

- `$ cut -d ":" -f 1,5 /etc/passwd` [*Invio*]

  Emette il primo e il quinto campo del file '`/etc/passwd`'. Per leggere correttamente il file, viene anche definito il tipo di separatore ('`:`'). In pratica, viene visualizzato il nominativo e il nome completo degli utenti.

22.7.2 Utilizzo di «paste»

Il programma di servizio '**paste**'[25] emette attraverso lo standard output l'unione, riga per riga, dei file indicati come argomento (in loro mancanza prende lo standard input). Le righe dei file vengono prese in ordine sequenziale e unite separandole con un carattere di tabulazione. Al termine delle nuove righe ottenute, viene aggiunto il codice di interruzione di riga.

```
paste [opzioni] [file...]
```

Tabella 22.29. Alcune opzioni.

Opzione	Descrizione
`-s` `--serial`	In questo caso viene utilizzato un solo file alla volta e tutte le sue righe vengono unite in un'unica riga.
`-d` ***elenco_delimitatori*** `--delimiters` ***elenco_delimitatori***	Viene utilizzato l'elenco di delimitatori fornito, invece di utilizzare la tabulazione per separare le righe riunite. Quando l'elenco di delimitatori viene esaurito, si ricomincia a usare il primo.

22.7.3 Utilizzo di «join»

Il programma di servizio '**join**'[26] genera un file contenente le righe che hanno chiavi identiche nei due file indicati tra gli argomenti. I due file devono essere già ordinati in base alle chiavi che si vogliono prendere in considerazione per la selezione.

```
join [opzioni] file1 file2
```

Si veda eventualmente il documento *info join* oppure la pagina di manuale *join(1)*.

22.8 Conversione

Alcuni programmi consentono di convertire il contenuto di un file, operando a livello di byte. La situazione più comune riguarda l'espansione dei caratteri di tabulazione (ovvero la loro trasformazione in caratteri spazio normali, nella quantità necessaria a mantenere le distanze previste) e, nel senso opposto, la conversione inversa per ridurre la dimensione complessiva del file.

22.8.1 Utilizzo di «expand»

Il programma di servizio '**expand**'[27] emette attraverso lo standard output una trasformazione del contenuto dei file forniti come argomento, oppure di quanto proviene dallo standard input, in cui i simboli di tabulazione sono trasformati in spazi veri e propri. Se

non viene specificato diversamente attraverso le opzioni, gli stop di tabulazione si intendono ogni otto caratteri.

`expand` [*opzioni*] [*file*...]

Tabella 22.30. Alcune opzioni.

Opzione	Descrizione
`-t` *tab1*[, *tab2*]... `--tabs=`*tab1*[, *tab2*]...	Permette di specificare una tabulazione diversa da quella predefinita (ogni otto colonne). Se viene specificato solo un numero, si intende una tabulazione ogni ***n*** colonne, altrimenti, si intende una sequenza di stop di tabulazione, in cui la prima colonna corrisponde al numero zero. Una volta esaurito l'elenco, gli stop di tabulazione successivi vengono sostituiti con uno spazio singolo.
`-`***n***	Questa è una variante dell'opzione '**`-t`**', in cui si mette direttamente il numero corrispondente alla tabulazione che si vuole avere.
`-i` `--initial`	Converte solo i caratteri di tabulazione iniziali, cioè quelli che precedono caratteri diversi da spazio, o che precedono altri caratteri di tabulazione.

I comandi mostrati sotto sono equivalenti: servono tutti a espandere i caratteri di tabulazione del file '`pippo.txt`' generando il file '`pippo1.txt`', utilizzando intervalli di tabulazione ogni otto colonne. Il valore è stato specificato per completezza, dal momento che un intervallo di otto colonne è quello predefinito.

```
$ expand -8 pippo.txt > pippo1.txt [Invio]
$ expand -t 8 pippo.txt > pippo1.txt [Invio]
$ expand --tabs=8 pippo.txt > pippo1.txt [Invio]
```

22.8.2 Utilizzo di «unexpand»

Il programma di servizio '**unexpand**'[28] emette attraverso lo standard output una trasformazione del contenuto dei file forniti come argomento, oppure dello standard input in loro mancanza, in cui gli spazi sono trasformati in caratteri di tabulazione per quanto possibile. Se non viene specificato diversamente attraverso le opzioni, gli stop di tabulazione si intendono ogni otto caratteri. Normalmente, il programma trasforma solo gli spazi iniziali.

`unexpand` [*opzioni*] [*file*...]

Tabella 22.31. Alcune opzioni.

Opzione	Descrizione
`-t` *tab1*[, *tab2*]... `--tabs=`*tab1*[, *tab2*]...	Permette di specificare una tabulazione diversa da quella predefinita (ogni otto colonne). Se viene specificato solo un numero, si intende una tabulazione ogni ***n*** colonne, altrimenti, si intende una sequenza di stop di tabulazione, in cui la prima colonna corrisponde al numero zero. Una volta esaurito l'elenco, non vengono fatte altre trasformazioni.
`-a` `--all`	Non si limita a trasformare solo gli spazi iniziali.

22.9 Traslitterazione o cancellazione di caratteri

Esiste un programma di servizio che spesso passa inosservato, ma è molto importante nell'elaborazione di file a livello di byte singolo (o di carattere). Si tratta di '**`tr`**',[29] (*translate*) il cui obiettivo fondamentale è quello di convertire un insieme di caratteri (o di simboli) in un altro insieme, ma consente anche l'eliminazione di alcuni caratteri

oppure solo di quelli doppi. Dal momento che con '**`tr`**' è necessario distinguere tra situazioni differenti, è opportuno descrivere separatamente la sua sintassi. L'elenco di modelli sintattici che viene mostrato è semplificato rispetto alle possibilità effettive di '**`tr`**'; inoltre si deve considerare che l'input proviene dallo standard input e l'output viene emesso attraverso lo standard output.

Modello sintattico	Descrizione
`tr` ***stringa1 stringa2***	Questo rappresenta la situazione comune, in cui l'insieme di caratteri indicato nella prima stringa viene sostituito con l'insieme dei caratteri della seconda stringa.
`tr -s` ***stringa1*** `tr --squeeze-repeats` ***stringa1***	Con l'opzione '**`-s`**' (ovvero '**`--squeeze-repeats`**'), si intendono eliminare i doppioni, relativi ai caratteri indicati nella stringa.
`tr -d` [`-c`] ***stringa1*** `tr --delete` [`--complement`] ↩ ↪ ***stringa1***	Con l'opzione '**`-d`**' (ovvero '**`--delete`**'), si intendono eliminare i caratteri indicati nella stringa. Se si usa anche l'opzione '**`-c`**' ('**`--complement`**'), i caratteri che vengono eliminati sono tutti esclusi quelli della stringa indicata.
`tr -d -s` ***stringa1 stringa2*** `tr --delete` ↩ ↪ `--squeeze-repeats` ↩ ↪ ***stringa1 stringa2***	Se alla cancellazione si aggiunge l'opzione di eliminazione dei doppioni, le cose vanno diversamente: prima vengono eliminati i caratteri corrispondenti a quelli della prima stringa; quindi vengono eliminati i doppioni dei caratteri appartenenti alla seconda stringa.

22.9.1 Rappresentazione dei caratteri in una stringa di «tr»

Le stringhe utilizzate come argomenti di '**`tr`**' vanno scritte secondo una sintassi particolare, che assomiglia vagamente alle espressioni regolari. In generale, ogni carattere (lettera, numero, simbolo) rappresenta esattamente se stesso, salvo le eccezioni che vengono descritte qui.

Sono ammissibili delle sequenze di escape formate da una barra obliqua inversa seguita da un carattere o da un numero che deve essere inteso in modo ottale. La tabella 22.33 elenca queste sequenze di escape (si veda anche la sezione 47.7.5).

Tabella 22.33. Sequenze di escape per le stringhe di '**`tr`**'.

Codice	Descrizione
`\a`	<*BEL*>.
`\b`	<*BS*>.
`\f`	<*FF*>.
`\n`	<*LF*>.
`\r`	<*CR*>.
`\t`	<*HT*> (tabulazione normale).
`\v`	<*VT*>.
`\`***nnn***	Il carattere corrispondente al codice ottale indicato.
`\\`	Una barra obliqua inversa singola.

Possono essere indicati degli intervalli di caratteri, attraverso la notazione '***m*-*n***'. Il carattere iniziale, ***m***, deve essere precedente a quello finale, in base alla sequenza stabilita dalla codifica a cui si fa riferimento. A titolo di esempio, l'intervallo '**0-4**', è equivalente alla sequenza di caratteri '**01234**'.

È possibile indicare una serie di caratteri ripetuti, attraverso una notazione particolare: '**[*x**n]**'. Qui le parentesi quadre fanno parte della notazione: ***x*** è il carattere da ripetere ***n*** volte. Se si omette il numero, si intende una quantità indefinitamente grande. È probabile che non sia conveniente l'uso di questa forma, anche per non complicare inutilmente l'interpretazione umana degli argomenti di '**tr**'; tuttavia è necessario conoscerne l'esistenza, per poter leggere gli script realizzati da altri.

Possono essere indicate delle classi di caratteri, in modo simile alle espressioni regolari: '**[:*classe*:]**'. Le classi utili nella traduzione da un insieme all'altro sono solo '**lower**' e '**upper**', allo scopo di convertire le lettere minuscole in maiuscole, oppure per fare l'inverso. Tutte le altre classi possono servire solo per la cancellazione dei caratteri, dal momento che non si espandono in un modo ordinato e prevedibile.

In teoria, è ammissibile anche la notazione delle classi di equivalenza: '**[=*x*=]**', che però, allo stato attuale, nella realizzazione GNU di '**tr**' non serve a molto, dal momento che si traduce semplicemente nella stessa lettera indicata (***x***).

22.9.2 Traslitterazione

La traslitterazione è l'operazione più semplice da realizzare con '**tr**'; basta l'indicazione delle stringhe: quella di origine e quella di destinazione. Vengono mostrati alcuni esempi.

```
$ tr abc def < primo.txt > secondo.txt [Invio]
```

Questo esempio mostra la lettura del file '`primo.txt`', che viene elaborato da '**tr**' in modo da trasformare ogni lettera «a» in «d», ogni lettera «b» in «e» e ogni lettera «c» in «f». Il risultato viene salvato nel file '`secondo.txt`'.

```
$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ ↩
↪< primo.txt > secondo.txt [Invio]

$ tr a-z A-Z < primo.txt > secondo.txt [Invio]

$ tr '[:lower:]' '[:upper:]' < primo.txt > secondo.txt [Invio]
```

Questi tre esempi sono molto simili: quello che si vuole ottenere è la conversione delle lettere minuscole in maiuscole. Di sicuro, il modo più corretto per raggiungere questo risultato è quello di utilizzare l'ultima delle tre forme, dal momento che così si dovrebbe garantire la compatibilità con le proprie convenzioni locali, includendo correttamente anche le lettere accentate (che qui non sono state mostrate).

22.9.3 Cancellazione e compattamento

La cancellazione serve solo per eliminare dei caratteri, senza convertirli in qualche modo. Il comando seguente può essere utile per convertire un file di testo in cui il codice di interruzione di riga è in stile Dos, ovvero è composto dalla sequenza <*CR*><*LF*>.

```
$ tr -d '\r' < primo.txt > secondo.txt [Invio]
```

In questo modo si elimina solo il codice <*CR*>, rappresentato dalla sequenza di escape '**\r**', ottenendo di lasciare solo i codici <*LF*>, adatti nella maggior parte dei sistemi Unix.[30]

```
$ tr -s '\n' < primo.txt > secondo.txt [Invio]
```

Con questo comando si vogliono eliminare le righe vuote multiple; tuttavia, non vengono eliminate le righe che sono semplicemente bianche, intese come quelle che contengono degli spazi.

```
$ tr -c -s '[a-zA-Z0-9]' '[ *]' ↩
↪< primo.txt > secondo.txt [Invio]
```


Questo esempio mostra l'uso dell'opzione di complemento ('**-c**'). In pratica, si vogliono identificare nella prima stringa tutti i caratteri che non siano alfanumerici (escludendo le lettere accentate), sostituendoli con un carattere spazio. Per indicare tanti caratteri spazio quanti sono necessari nella seconda stringa, si utilizza la notazione speciale '**[*]**' che ripete lo spazio in modo indefinito. Infine, gli spazi multipli vengono semplicemente eliminati.

```
$ tr -c -s '[a-zA-Z0-9]' '[\n*]' ↩
↪< primo.txt > secondo.txt [Invio]
```

Il comando appena mostrato si comporta come nell'esempio precedente, solo che invece di usare lo spazio per sostituire ciò che non è un carattere alfanumerico, si utilizza il codice <*LF*>, corrispondente al codice di interruzione di riga. Lo scopo è quello di individuare tutte le parole del testo e di ottenerne l'elenco, dove ognuna occupa una riga separata.

```
$ tr -c -s '[:alnum:]' '[\n*]' < primo.txt > secondo.txt [Invio]
```

Questa variante precisa meglio l'intenzione di selezionare tutti i caratteri non alfanumerici, perché la stringa di cui si fa il complemento contiene l'indicazione della classe '**alnum**', che comprende anche le lettere accentate della propria localizzazione.

22.10 Controlli sommari

Alcuni programmi si occupano di calcolare valori di vario genere in base al contenuto dell'input. Può trattarsi del conteggio di elementi determinati o del calcolo di codici di controllo (*checksum*).

22.10.1 Utilizzo di «sum»

Il programma di servizio '**sum**'[31] calcola un codice di controllo a 16 bit per ogni file fornito negli argomenti. Emette attraverso lo standard output il valore ottenuto insieme alla dimensione (arrotondata) in blocchi. Il valore predefinito della dimensione dei blocchi è di 1024 byte.

```
sum [opzioni] [file...]
```

Questo programma si considera superato e al suo posto si preferisce utilizzare '**cksum**'

Tabella 22.34. Alcune opzioni.

Opzione	Descrizione
`-r`	Utilizza l'algoritmo predefinito (compatibile con BSD).
`-s` `--sysv`	Utilizza l'algoritmo compatibile con System V. In tal caso, i blocchi hanno una dimensione di 512 byte.

22.10.2 Utilizzo di «cksum»

Il programma di servizio '**cksum**'[32] calcola un codice di controllo CRC (*Cyclic redundancy check*) per ogni file fornito negli argomenti, oppure dello standard input in mancanza di questi:

```
cksum [opzioni] [file...]
```

Non utilizza opzioni, tranne quelle comuni dei programmi GNU.

22.10.3 Utilizzo di «md5sum»

Il programma di servizio '**md5sum**'[33] calcola un codice di controllo MD5 (*Message digest*) a 128 bit per ogni file fornito negli argomenti, oppure verifica la corrispondenza di una serie di codici di controllo accumulati precedentemente in un file, con i file relativi:

```
md5sum [opzioni] [file...]
```

In condizioni normali, '**md5sum**' emette una serie di righe nella forma:

```
firma file
```

Per esempio, una cosa del genere:

```
fdbf0c571fb4942a6c505d732e163876  a2ps.1.gz
f2c766c141c6e5bb55c8edf6ce4ecba6  ab.1.gz
00169ba95302aca74597f000b61c3255  access.1.gz
69cf0ef0436aff6830a8a8a11b53b27b  addftinfo.1.gz
```

Questa informazione può essere conservata per verificare in seguito se gli stessi file corrispondono sempre agli originali, oppure se sono stati danneggiati o manomessi. La verifica può essere manuale (visiva), oppure può essere lo stesso '**md5sum**' a verificarla, utilizzando per questo l'opzione '**-c**'.

Tabella 22.36. Alcune opzioni.

Opzione	Descrizione
`-c` *file*	Utilizzando questa opzione, negli argomenti di '**md5sum**' si può indicare solo un file, costituito dall'elenco di firme abbinate ai file corrispondenti: vengono ricalcolate le firme di questi file e viene verificata la corrispondenza con le firme già annotate. In caso di differenza, viene emessa una segnalazione di errore.
`-v`	Genera maggiori informazioni.

Segue la descrizione di alcuni esempi.

- `$` **`md5sum *.txt > firme`** [*Invio*]

  Raccoglie le firme MD5 di tutti i file che terminano con l'estensione '`.txt`'.

- `$` **`md5sum -c firme`** [*Invio*]

  Controlla tutti i file elencati nel file '`firme`' per verificare che il contenuto di questi non sia stato alterato in alcun modo.

22.10.4 Utilizzo di «sha1sum»

Il programma di servizio '**sha1sum**'[34] calcola un codice di controllo SHA1 a 160 bit per ogni file fornito negli argomenti, oppure verifica la corrispondenza di una serie di codici di controllo accumulati precedentemente in un file, con i file relativi:

```
sha1sum [opzioni] [file...]
```

```
sha1sum [opzioni] -c file
```

In condizioni normali il programma serve a calcolare il codice di controllo dei file indicati alla fine della riga di comando, generando un risultato simile a quello dell'esempio seguente:

`$` **`sha1sum /bin/*sh`** [*Invio*]

```
13c36c7b1327b2bf4bc692e27e55125f6335ea82  /bin/ash
f4e6f05cafcffa4c43218f5b8c9e55a056cea459  /bin/bash
13c36c7b1327b2bf4bc692e27e55125f6335ea82  /bin/dash
20e6f3f8e2705e281be4892498550fd4a8c7c3e0  /bin/fdflush
f4e6f05cafcffa4c43218f5b8c9e55a056cea459  /bin/rbash
f4e6f05cafcffa4c43218f5b8c9e55a056cea459  /bin/sh
```

Questa informazione può essere conservata per verificare in seguito se gli stessi file corrispondono sempre agli originali, oppure se sono stati danneggiati o manomessi. La verifica può essere manuale (visiva), oppure può essere lo stesso programma a farla, utilizzando per questo l'opzione '**-c**', ma in tal caso si fornisce un solo file, che corrisponde a quello contenente l'elenco accumulato in precedenza. Nell'esempio seguente si suppone di avere salvato il controllo precedente nel file '`elenco.sha1sum`':

`$` **`sha1sum -c elenco.sha1sum`** [*Invio*]


```
/bin/ash: OK
/bin/bash: OK
/bin/dash: OK
/bin/fdflush: OK
/bin/rbash: OK
/bin/sh: OK
```

Tabella 22.39. Alcune opzioni.

Opzione	Descrizione
`-c` *file* `--check` *file*	Si richiede di eseguire la verifica dei codice di controllo in base all'elenco di firme SHA1 accumulate in precedenza nel file.
`-w` `--warn`	Mostra informazioni diagnostiche nel caso il file da usare per il confronto contenga delle righe imperfette.

22.11 Lettura di file binari

Per poter leggere il contenuto di file che non sono soltanto file di testo, diventa necessario trasformare alcuni o tutti i byte in qualcosa di leggibile, come può essere in esadecimale o in ottale.

22.11.1 Utilizzo di «hexdump» o di «hd»

Il programma di servizio '**hexdump**', o '**hd**',[35] consente di visualizzare il contenuto di un file binario attraverso una qualche forma di trasformazione utile per la lettura umana:

```
hexdump [opzioni] [file...]
```

```
hd [opzioni] [file...]
```

I due modelli sintattici mostrano l'uso di questo programma attraverso nomi diversi; in particolare, l'uso del nome '**hd**' implica automaticamente la presenza dell'opzione '**-C**'. Se si omette l'indicazione dei file nella riga di comando, il programma utilizza per questo lo standard input.

Tabella 22.40. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-b`	*byte*	Mostra una traduzione del contenuto del file in ottale, separando i codici in ottale, byte per byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-c`	*character*	Mostra una traduzione del contenuto del file in modo normale, quando possibile, altrimenti usa la notazione ottale. I byte vengono separati uno a uno. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-C`	*canonical*	Mostra il risultato in forma «canonica», pari all'uso del comando con il nome '**hd**'.
`-d`	*decimal*	Mostra una traduzione del contenuto del file in forma decimale, dove ogni valore corrisponde a una coppia di byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-o`	*octal*	Mostra una traduzione del contenuto del file in forma ottale, a coppie di byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-x`	*exadecimal*	Mostra una traduzione del contenuto del file in forma esadecimale, a coppie di byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.

Opzione	Significato mnemonico	Descrizione
`-l` ***n***	*length*	Legge solo i primi ***n*** byte.
`-s` ***n***	*skip*	Salta i primi ***n*** byte a partire dall'inizio del file.

Vengono mostrati alcuni esempi, senza spiegazione, che prendono in considerazione lo stesso file di partenza:

```
$ hexdump /bin/arch [Invio]

0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000010 0002 0003 0001 0000 8300 0804 0034 0000
0000020 0710 0000 0000 0000 0034 0020 0007 0028
0000030 0018 0017 0006 0000 0034 0000 8034 0804
0000040 8034 0804 00e0 0000 00e0 0000 0005 0000
…
0000aa0 0004 0000 0000 0000 0001 0000 0003 0000
0000ab0 0000 0000 0000 0000 0658 0000 00b5 0000
0000ac0 0000 0000 0000 0000 0001 0000 0000 0000
0000ad0
```

```
$ hd /bin/arch [Invio]

00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  00 83 04 08 34 00 00 00  |............4...|
00000020  10 07 00 00 00 00 00 00  34 00 20 00 07 00 28 00  |........4. ...(.|
00000030  18 00 17 00 06 00 00 00  34 00 00 00 34 80 04 08  |........4...4...|
00000040  34 80 04 08 e0 00 00 00  e0 00 00 00 05 00 00 00  |4...à...à.......|
…
00000aa0  04 00 00 00 00 00 00 00  01 00 00 00 03 00 00 00  |................|
00000ab0  00 00 00 00 00 00 00 00  58 06 00 00 b5 00 00 00  |........X...µ...|
00000ac0  00 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................|
00000ad0
```

```
$ hexdump -b /bin/arch [Invio]

0000000 177 105 114 106 001 001 001 000 000 000 000 000 000 000 000 000
0000010 002 000 003 000 001 000 000 000 000 203 004 010 064 000 000 000
0000020 020 007 000 000 000 000 000 000 064 000 040 000 007 000 050 000
0000030 030 000 027 000 006 000 000 000 064 000 000 000 064 200 004 010
0000040 064 200 004 010 340 000 000 000 340 000 000 000 005 000 000 000
…
0000aa0 004 000 000 000 000 000 000 000 001 000 000 000 003 000 000 000
0000ab0 000 000 000 000 000 000 000 000 130 006 000 000 265 000 000 000
0000ac0 000 000 000 000 000 000 000 000 001 000 000 000 000 000 000 000
0000ad0
```

```
$ hexdump -c /bin/arch [Invio]

0000000 177   E   L   F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0
0000010 002  \0 003  \0 001  \0  \0  \0  \0 203 004  \b   4  \0  \0  \0
0000020 020  \a  \0  \0  \0  \0  \0  \0   4  \0      \0  \a  \0   (  \0
0000030 030  \0 027  \0 006  \0  \0  \0   4  \0  \0  \0   4 200 004  \b
0000040   4 200 004  \b   à  \0  \0  \0   à  \0  \0  \0 005  \0  \0  \0
…
0000aa0 004  \0  \0  \0  \0  \0  \0  \0 001  \0  \0  \0 003  \0  \0  \0
0000ab0  \0  \0  \0  \0  \0  \0  \0  \0   X 006  \0  \0   µ  \0  \0  \0
0000ac0  \0  \0  \0  \0  \0  \0  \0  \0 001  \0  \0  \0  \0  \0  \0  \0
0000ad0
```

```
$ hexdump -d /bin/arch [Invio]

0000000   17791   17996   00257   00001   00000   00000   00000   00000
0000010   00002   00003   00001   00000   33536   02052   00052   00000
0000020   01808   00000   00000   00000   00052   00032   00007   00040
0000030   00024   00023   00006   00000   00052   00000   32820   02052
0000040   32820   02052   00224   00000   00224   00000   00005   00000
…
0000aa0   00004   00000   00000   00000   00001   00000   00003   00000
0000ab0   00000   00000   00000   00000   01624   00000   00181   00000
0000ac0   00000   00000   00000   00000   00001   00000   00000   00000
0000ad0
```

```
$ hexdump -o /bin/arch [Invio]

0000000  042577  043114  000401  000001  000000  000000  000000  000000
0000010  000002  000003  000001  000000  101400  004004  000064  000000
0000020  003420  000000  000000  000000  000064  000040  000007  000050
0000030  000030  000027  000006  000000  000064  000000  100064  004004
0000040  100064  004004  000340  000000  000340  000000  000005  000000
…
0000aa0  000004  000000  000000  000000  000001  000000  000003  000000
0000ab0  000000  000000  000000  000000  003130  000000  000265  000000
0000ac0  000000  000000  000000  000000  000001  000000  000000  000000
0000ad0
```

```
$ hexdump -x /bin/arch [Invio]

0000000    457f    464c    0101    0001    0000    0000    0000    0000
0000010    0002    0003    0001    0000    8300    0804    0034    0000
0000020    0710    0000    0000    0000    0034    0020    0007    0028
0000030    0018    0017    0006    0000    0034    0000    8034    0804
0000040    8034    0804    00e0    0000    00e0    0000    0005    0000
…
0000aa0    0004    0000    0000    0000    0001    0000    0003    0000
0000ab0    0000    0000    0000    0000    0658    0000    00b5    0000
0000ac0    0000    0000    0000    0000    0001    0000    0000    0000
0000ad0
```

La pagina di manuale *hexdump(1)* riporta anche altre opzioni che qui non sono state indicate; in particolare, è possibile programmare il risultato dell'elaborazione attraverso le opzioni '**`-e`**' e '**`-f`**'.

22.11.2 Utilizzo di «hexcat»

Il programma di servizio '**`hexcat`**',[36] consente di visualizzare il contenuto di uno o più file in forma esadecimale. Non prevede alcuna opzione:

```
hexcat [file…]
```

L'esempio seguente mostra il funzionamento di '**`hexcat`**':

```
$ hexcat /bin/true [Invio]

00000000 - 7f 45 4c 46  01 01 01 00  00 00 00 00  00 00 00 00  .ELF............
00000010 - 02 00 03 00  01 00 00 00  60 89 04 08  34 00 00 00  ........‘...4...
00000020 - b8 29 00 00  00 00 00 00  34 00 20 00  07 00 28 00  ¸)......4. ...(.
00000030 - 18 00 17 00  06 00 00 00  34 00 00 00  34 80 04 08  ........4...4...
00000040 - 34 80 04 08  e0 00 00 00  e0 00 00 00  05 00 00 00  4...............
00000050 - 04 00 00 00  03 00 00 00  14 01 00 00  14 81 04 08  ................
00000060 - 14 81 04 08  13 00 00 00  13 00 00 00  04 00 00 00  ................
00000070 - 01 00 00 00  01 00 00 00  00 00 00 00  00 80 04 08  ................
…
…
00002d40 - 00 00 00 00  00 00 00 00  20 00 00 00  00 00 00 00  ........ .......
00002d50 - 01 00 00 00  03 00 00 00  00 00 00 00  00 00 00 00  ................
00002d60 - 00 29 00 00  b5 00 00 00  00 00 00 00  00 00 00 00  .)..µ...........
00002d70 - 01 00 00 00  00 00 00 00                           ........
```

22.12 Differenze tra i file

Spesso esiste la necessità di confrontare il contenuto tra dei file per verificare se esistono delle differenze, ma soprattutto per conoscere quali sono, quando queste non sono troppe. Se le differenze tra i due file sono in numero ragionevolmente contenuto, si può generarne un rapporto, in modo da poter ottenere uno dei due file a partire dall'altro, assieme a tale elenco di variazioni.

Questo rapporto sulle differenze, definito prevalentemente *patch*, si applica a un file, o a una serie di file, per ottenere altrettanti file aggiornati.

Esistono tanti modi di costruire un file di differenze. Si distinguono in particolare due situazioni: i file di testo e gli altri. Si può comprendere che in un file di testo, tipicamente un sorgente di un programma, i cambiamenti avvengano a livello di righe, nel senso che se ne possono aggiungere, togliere e modificare. In un file binario invece, non avendo il riferimento delle righe, il problema è più complesso. La gestione delle differenze tra i file riguarda prevalentemente i file di testo normale, cosa di cui si vuole trattare in questo capitolo.

22.12.1 Creazione di un file di differenze con «diff»

Il programma più importante per analizzare le differenze tra due file di testo è '**`diff`**'.[37] Può funzionare con diverse modalità, per determinare semplicemente se una coppia di file è identica o meno, oppure per indicare le differenze che ci sono tra i due, con maggiore o minore dettaglio di informazioni al riguardo. La sintassi sintetica di questo programma è molto semplice.

```
diff [opzioni] file_1 file_2
```

Il risultato del confronto dei file viene emesso attraverso lo standard output.

22.12.1.1 Regolazione della sensibilità di «diff»

Quando si confrontano file di testo, può darsi che alcuni tipi di differenze non siano da considerare, come per esempio l'aggiunta di spazi alla fine di una riga, o l'inserzione di righe vuote addizionali. Inoltre, si può desiderare di conoscere semplicemente se esiste una qualche differenza, senza entrare troppo nel dettaglio. Questa sensibilità alle differenze viene definita attraverso l'uso di opzioni apposite. Le più importanti sono elencate nella tabella successiva.

Tabella 22.49. Opzioni per il controllo della sensibilità di '**diff**'.

Opzione	Descrizione
`-q` `--brief`	Rapporto sommario: attraverso questa opzione si richiede a '**diff**' di informare semplicemente sull'esistenza di differenze tra due file, senza l'indicazione esplicita di queste.
`-i` `--ignore-case`	Maiuscole e minuscole: attraverso questa opzione si può richiedere a '**diff**' di ignorare la differenza tra maiuscole e minuscole.
`-b` `--ignore-space-change`	Spaziatura orizzontale ridondante: questa opzione permette di fare ignorare a '**diff**' le differenze dovute a una diversa spaziatura orizzontale del testo. Questo riguarda quindi, sia il carattere spazio, <*SP*>, sia il carattere di tabulazione, <*HT*>.
`-w` `--ignore-all-space`	Spaziatura orizzontale: questa opzione permette di fare ignorare completamente a '**diff**' la presenza degli spazi.
`-B` `--ignore-blank-lines`	Spaziatura verticale: questa opzione permette di fare ignorare a '**diff**' le differenze dovute alla presenza o assenza di righe vuote. Deve trattarsi però di righe completamente vuote, cioè composte esclusivamente dal codice di interruzione di riga.

Con l'opzione '**-i**' le due righe seguenti sono considerate equivalenti:

```
Chi va piano,

chi va PIANO,
```

Con l'opzione '**-b**', le due righe seguenti sono considerate equivalenti:

```
va sano e va lontano

va sano  e   va lontano
```

Con l'opzione '**-w**', le due righe seguenti sono equivalenti:

```
vasano e va lon   tano

va sano  e  va lontano
```

22.12.1.2 Confronto binario o testuale

Prima di iniziare un confronto tra due file, '**diff**' verifica che si tratti di file di testo in base al contenuto di alcune righe iniziali. Se '**diff**' incontra il carattere <*NUL*>, a meno che siano state usate opzioni particolari in senso contrario, assume che si tratti di un file binario e verifica semplicemente se i file sono identici.

Tabella 22.53. Opzioni per il tipo di confronto.

Opzione	Descrizione
`--binary`	Il confronto binario può essere imposto attraverso questa opzione e ciò che si ottiene è solo la verifica sull'identicità dei file. Se la prima parte di uno dei file da confrontare contiene il carattere <*NUL*>, '**diff**' assume implicitamente che debba essere eseguito un confronto binario.

Opzione	Descrizione
`-a` `--text`	Il confronto testuale, cioè quello normale, può essere imposto con questa opzione anche in presenza di caratteri <*NUL*> iniziali, per esempio quando si vogliono confrontare file generati da programmi per l'elaborazione testi che sfruttano quel carattere per scopi particolari.

22.12.1.3 Differenze senza contesto

Il funzionamento normale di '**diff**' prevede l'emissione attraverso lo standard output dell'indicazione delle sole differenze tra i file, secondo il formato seguente:

```
comando
< riga_primo_file
< riga_primo_file
< …
---
> riga_secondo_file
> riga_secondo_file
> …
```

In questo tipo di notazione, è il «comando» a stabilire l'operazione da compiere. Il comando si compone di tre parti: il numero di una riga, o di un intervallo di righe del primo file; una lettera che definisce l'operazione da compiere; il numero di una riga, o di un intervallo di righe del secondo file.

```
righe_file_1 azione righe_file_2
```

Si distinguono le tre azioni seguenti.

- **Aggiunta**

  ```
  posizione_file_1 a righe_file_2
  ```

  Indica che per ottenere il secondo file a partire dal primo, occorre aggiungere a questo le righe indicate a destra dell'azione, dopo la posizione indicata a sinistra. Per esempio, '**5a6,8**' significa che per ottenere il secondo file occorre aggiungere al primo le righe dalla sesta all'ottava del secondo, dopo la quinta riga del primo file.

- **Sostituzione**

  ```
  righe_file_1 c righe_file_2
  ```

  Indica di sostituire le righe del primo file, indicate a sinistra dell'azione, con quelle del secondo file indicate a destra.

- **Cancellazione**

  ```
  righe_file_1 d posizione_file_2
  ```

  Indica che per ottenere il secondo file a partire dal primo, occorre eliminare dal primo le righe indicate a sinistra dell'azione. L'indicazione della posizione del secondo file serve solo per completezza, a specificare il punto in cui tali righe mancano. In pratica, l'azione '**d**' è l'inverso dell'azione '**a**'.

Quando si vogliono distribuire file di differenze (o delle *patch*, se si preferisce il termine) per consentire ad altri di ottenere degli aggiornamenti da un file di partenza, è sconsigliabile l'utilizzo di questo formato, benché si tratti di quello predefinito per '**diff**', secondo lo standard POSIX.

Per verificare in pratica il funzionamento di '**diff**' in modo da ottenere l'indicazione delle differenze tra due file senza informazioni sul contesto, viene proposto il confronto tra i due file seguenti:

```
Chi va piano,
va sano
e va lontano
```

```
chi va piano,
va     sano
e va lontano
```

I nomi dei due file siano rispettivamente: ‘`primo`’ e ‘`secondo`’.

```
$ diff primo secondo [Invio]

1,2c1,2
< Chi va piano,
< va sano
---
> chi va piano,
> va     sano
```

In pratica, le prime due righe del primo file vanno sostituite con le prime due del secondo, mentre la terza riga è la stessa in entrambi i file.

```
$ diff -i primo secondo [Invio]

2c2
< va sano
---
> va     sano
```

In questo caso, utilizzando l’opzione ‘**`-i`**’, si vogliono ignorare le differenze tra lettere maiuscole e minuscole, pertanto risulta diversa solo la seconda riga.

```
$ diff -b primo secondo [Invio]

1c1
< Chi va piano,
---
> chi va piano,
```

In questo caso, utilizzando l’opzione ‘**`-b`**’ si vogliono ignorare le differenze dovute a un uso differente delle spaziature tra le parole, pertanto risulta diversa solo la prima riga.

22.12.1.4 Formato contestuale standard

Il funzionamento normale di ‘**`diff`**’ prevede l’emissione attraverso lo standard output dell’indicazione delle sole differenze tra i file, ma ciò è generalmente poco adatto alla distribuzione di file di differenze. Per questo è preferibile utilizzare un formato che, assieme alle modifiche, inserisca anche alcune righe di riferimento aggiuntive. In questo modo, il programma che deve applicare le modifiche, può agire anche se il contenuto del file sul quale vengono applicate ha subito piccoli spostamenti. Si ottiene un formato contestuale standard quando si utilizza l’opzione seguente:

```
-c | -C righe | --context[=righe]
```

Se viene indicato il numero di righe, si intende che venga utilizzato almeno quel numero di righe di riferimento contestuale. Se questo valore non viene indicato, si intende che siano tre. Il minimo perché il programma ‘**`patch`**’ possa eseguire il suo compito è di due righe contestuali.

Il risultato di una comparazione contestuale standard è preceduto da due righe di intestazione contenenti l’indicazione dei due file.

```
*** file_1  data_di_modifica_del_primo_file
--- file_2  data_di_modifica_del_secondo_file
```

Successivamente appaiono i blocchi delle differenze, strutturati nel modo seguente:


```
***************
*** righe_primo_file ****
  riga_primo_file
  riga_primo_file
  …
--- righe_corrispondenti_secondo_file ----
  riga_secondo_file
  riga_secondo_file
  …
```

Si deve osservare che le righe vengono indicate a partire dalla terza colonna, lasciando cioè due spazi dall’inizio. La prima colonna viene utilizzata per indicare il ruolo particolare di quella riga:

- se non appare alcun simbolo, si tratta di una riga di contesto che non risulta modificata;
- il punto esclamativo (‘**`!`**’) rappresenta un cambiamento tra i due file;
- il segno ‘**`+`**’ rappresenta una riga aggiunta nel secondo file che nel primo non esiste;
- il segno ‘**`-`**’ rappresenta una riga nel primo file che nel secondo risulta cancellata.

Per verificare in pratica il funzionamento di ‘**`diff`**’ in modo da utilizzare il formato contestuale standard, viene proposto il confronto tra i due file seguenti:

```
Chi va piano,
va sano
e va lontano
```

```
Chi va forte,
va alla morte;

chi va piano,
va     sano
e va lontano
```

I nomi dei due file siano rispettivamente: ‘`primo`’ e ‘`secondo`’.

```
$ diff -c primo secondo [Invio]

*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
! Chi va piano,
! va sano
  e va lontano
--- 1,6 ----
! Chi va forte,
! va alla morte;
!
! chi va piano,
! va     sano
  e va lontano
```

In breve, le prime tre righe del primo file vanno sostituite con le prime sei del secondo e l’unica riga in comune è l’ultima.

```
$ diff -c -i primo secondo [Invio]

*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
  Chi va piano,
! va sano
  e va lontano
--- 1,6 ----
+ Chi va forte,
+ va alla morte;
+
  chi va piano,
! va     sano
  e va lontano
```

In questo caso, vanno aggiunte le prime tre righe del secondo file, quindi si incontra una riga uguale, dal momento che non contano le

differenze tra lettere maiuscole e minuscole, infine viene sostituita una riga a causa della spaziatura orizzontale differente.

```
$ diff -b -i -c primo secondo [Invio]

*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
--- 1,6 ----
+ Chi va forte,
+ va alla morte;
+
  chi va piano,
  va     sano
  e va lontano
```

In questo caso, avendo indicato che non contano le differenze dovute alla diversa spaziatura orizzontale e all'uso delle maiuscole, le ultime tre righe del secondo file corrispondono esattamente al primo file. In questo modo, tali righe non sono state indicate nella parte che riguarda il primo file.

22.12.1.5 Formato contestuale unificato

A fianco del formato contestuale standard, si pone un altro tipo di indicazione delle modifiche, definito «unificato», il quale ha il vantaggio di essere più compatto, ma anche lo svantaggio di essere disponibile solo negli strumenti GNU. Per selezionare questo tipo di risultato si utilizza una delle opzioni seguenti.

```
-u | -U righe | --unified[=righe]
```

Se viene indicato il numero di righe, si intende che venga utilizzato almeno quel numero di righe di riferimento contestuale. Se questo valore non viene indicato, si intende che siano tre. Il minimo perché il programma '**patch**' possa eseguire il suo compito è di due righe contestuali.

Il risultato di una comparazione contestuale unificata è preceduto da due righe di intestazione contenenti l'indicazione dei due file.

```
--- file_1  data_di_modifica_del_primo_file
+++ file_2  data_di_modifica_del_secondo_file
```

Successivamente appaiono i blocchi delle differenze, strutturati nel modo seguente:

```
@@ -righe_primo_file +righe_secondo_file @@
  riga_di_uno_dei_file
  riga_di_uno_dei_file
  …
```

In modo simile al caso del formato contestuale standard, le righe sono riportate a partire dalla seconda colonna, lasciando il primo carattere libero per indicare l'operazione da compiere:

- le righe comuni a entrambi i file iniziano con un carattere spazio (<*SP*>);
- il segno '**+**' rappresenta l'inserimento di una riga, in quel punto, rispetto al contenuto del primo file;
- il segno '**-**' rappresenta una riga nel primo file che nel secondo risulta cancellata.

Per verificare in pratica il funzionamento di '**diff**' in modo da utilizzare il formato contestuale unificato, vengono proposti gli stessi esempi già visti nella sezione precedente:

```
$ diff -u primo secondo [Invio]

--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
-Chi va piano,
-va sano
+Chi va forte,
+va alla morte;
+
+chi va piano,
+va     sano
 e va lontano
```

In breve, le prime tre righe del primo file vanno sostituite con le prime sei del secondo e l'unica riga in comune è l'ultima.

```
$ diff -u -i primo secondo [Invio]

--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
+Chi va forte,
+va alla morte;
+
 Chi va piano,
-va sano
+va     sano
 e va lontano
```

In questo caso, vanno aggiunte le prime tre righe del secondo file, quindi si incontra una riga uguale, dal momento che non contano le differenze tra lettere maiuscole e minuscole, infine viene sostituita una riga a causa della spaziatura orizzontale differente.

```
$ diff -b -i -u primo secondo [Invio]

--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
+Chi va forte,
+va alla morte;
+
 Chi va piano,
 va sano
 e va lontano
```

In questo caso, avendo indicato che non contano le differenze dovute alla diversa spaziatura orizzontale e all'uso delle maiuscole, le ultime tre righe del secondo file corrispondono esattamente al primo file. In questo modo, tali righe non sono state indicate nella parte che riguarda il primo file.

22.12.1.6 Confronto dei file di due directory

Il programma '**diff**' è in grado di generare un file di differenze unico, dal confronto di tutti i file di una directory con altrettanti file con lo stesso nome contenuti in un'altra. per ottenere questo, è sufficiente indicare il confronto di due directory, invece che di due file.

Se si desidera continuare l'analisi anche nelle sottodirectory successive, si può utilizzare l'opzione seguente:

```
-r | --recursive
```

Si suppone che '`uno/`' e '`due/`' siano due sottodirectory della directory corrente nel momento in cui si esegue '**diff**':

```
$ diff -u uno due [Invio]
```

Ciò che si ottiene attraverso lo standard output è l'elenco delle modifiche dei vari file incontrati in entrambe le directory. Quello che segue è un estratto delle intestazioni in cui si vede in che modo sono indicati i file, assieme al loro percorso relativo:

```
…
--- uno/primo   Thu Mar  5 09:48:10 1998
+++ due/primo   Fri Mar  6 08:30:07 1998
…
--- uno/secondo Wed Mar  4 09:23:52 1998
+++ due/secondo Fri Mar  6 08:29:59 1998
…
```

Se all'esempio precedente si aggiunge l'opzione '**-r**', '**diff**' attraversa anche le sottodirectory contenute in quelle indicate nella riga di comando:

```
$ diff -u -r uno due [Invio]
```

22.12.1.7 Come si prepara un file di differenze

Quando si prepara un file di differenze, è opportuno usare un po' di accortezza per facilitare il lavoro di chi poi deve applicare queste modifiche. È il caso di distinguere due situazioni fondamentali: le differenze riferite a un file singolo e quelle relative a un intero ramo di directory.

Per prima cosa occorre decidere il tipo di formato: quello predefinito non è molto comodo perché non contiene le informazioni sui nomi dei file, mentre quello contestuale unificato dovrebbe essere il migliore. Tuttavia, quando si devono produrre file di differenze da utilizzare con strumenti strettamente POSIX, ci si deve accontentare del formato contestuale standard.

In generale, è importante l'ordine in cui si indicano i file o le directory tra gli argomenti di '**diff**': il primo dei due nomi rappresenta la situazione precedente, mentre il secondo quella nuova, ovvero l'aggiornamento verso cui si vuole andare. La situazione classica è quella in cui si modifica un file, ma prima di intervenire se ne salva una copia con la tipica estensione '`.orig`'. Si osservi l'esempio seguente:

```
$ diff -u prova.txt.orig prova.txt > prova.diff [Invio]
```

Il file '`prova.txt`' è stato modificato, ma prima di farlo ne è stata salvata una copia con il nome '`prova.txt.orig`'. Il comando genera un file di differenze tra '`prova.txt.orig`' e '`prova.txt`'.

Per realizzare un file di differenze di un ramo intero di directory, si interviene in modo simile: si fa una copia del ramo, si modifica quello che si vuole nei file del ramo che si intende debba contenere gli aggiornamenti, quindi si utilizza '**diff**':

```
$ diff -u -r prova.orig prova > prova.diff [Invio]
```

In questo caso, si intende fare riferimento al confronto tra le directory '`prova.orig/`' e '`prova/`'. Il file di differenze che si ottiene è unico per tutti i file che risultano modificati effettivamente.

22.12.2 Applicazione delle modifiche con «patch»

Il programma più adatto per applicare delle modifiche è '**patch**',[38] il quale di solito è in grado di determinare automaticamente il tipo di formato utilizzato e di saltare eventuali righe iniziali o finali aggiuntive. In pratica, con '**patch**' è possibile utilizzare file trasmessi come allegato nei messaggi di posta elettronica, senza doverli estrapolare.

```
patch [opzioni] < file_di_differenze
```

Il programma '**patch**' va usato preferibilmente, fornendo il file di differenze attraverso lo standard input, perché diversamente, la versione GNU ha una sintassi non perfettamente conforme allo standard POSIX. In questo modo, il file di differenze deve contenere l'indicazione del file da modificare, pertanto si può utilizzare soltanto un formato contestuale (compreso quello unificato).

In linea di massima, '**patch**' sovrascrive i file a cui si vogliono applicare delle modifiche, a meno che venga specificata un'opzione con la quale si richiede l'accantonamento di una copia della versione precedente. In questo caso, il file originale viene rinominato (in condizioni normali gli viene aggiunta l'estensione '`.orig`') e gli aggiornamenti vengono applicati a questo file ottenendone un altro con lo stesso nome di quello originale. Il programma '**patch**' cerca di applicare le modifiche anche quando il file di partenza non risulta perfettamente corrispondente a quanto indicato nel file di differenze. Se qualche blocco di modifiche non può essere applicato, questi vengono indicati in un file terminante con l'estensione '`.rej`' (*rejected*).

Tabella 22.68. Alcune opzioni.

Opzione	Descrizione
`-d` *directory*	Modifica la directory di lavoro prima di iniziare.
`-p`*n*	Elimina ***n*** barre oblique iniziali da un percorso.
`-o` *file_aggiornato*	Indica precisamente il file da ottenere applicando le modifiche.
`-b`	Mantiene una copia della versione precedente.
`-l`	Tratta come equivalenti le sequenze di spazi orizzontali.
`-r` *file_rigetti*	Indica precisamente il file che deve contenere gli errori.
`-R`	Applica le modifiche in modo inverso.
`-N`	Ignora le modifiche che sembrano essere già state applicate.

22.12.2.1 Definizione dei file da modificare e del file di differenze

In condizioni normali, precisamente quando si dispone di file di differenze in formato contestuale (standard o unificato), non è necessario fornire a '**patch**' il nome del file su cui intervenire per applicare le modifiche, perché questo è indicato all'interno del file che le contiene. Tuttavia, il formato predefinito lo impone e in ogni caso può essere utile indicare precisamente a '**patch**' il nome del file su cui intervenire.

```
patch [opzioni] file_originale [file_di_differenze]
```

Lo schema mostra semplicemente che è sufficiente accodare dopo le opzioni il nome del file originale al quale si vogliono applicare le modifiche (ma questo modello sintattico non è conforme allo standard POSIX). Le modifiche possono essere contenute in un file indicato come argomento successivo, oppure fornito attraverso lo standard input, come si fa di solito. In alternativa, il file di differenze può anche essere indicato in modo esplicito attraverso l'opzione '**-i**' (ovvero '**--input**'), mentre per il file da generare si può usare l'opzione '**-o**' (ovvero '**--output**').

```
patch [opzioni] [-i file_di_differenze] [-o file_da_generare] ↩
↪ [file_originale]
```

Gli esempi seguenti sono equivalenti e servono ad applicare al file '`prova`' le modifiche contenute nel file '`prova.diff`', sovrascrivendo il file '`prova`' stesso:

```
$ patch prova prova.diff [Invio]

$ patch prova < prova.diff [Invio]

$ patch -i prova.diff prova [Invio]

$ patch --input=prova.diff prova [Invio]
```

22.12.2.2 Definizione esplicita del formato del file di differenze

In alcune circostanze, può essere utile, o necessario, definire esplicitamente di quale tipo sia il formato del file di differenze. A questo proposito si utilizzano alcune opzioni:

Opzione	Descrizione
`-n` `--normal`	per indicare un formato normale;
`-c` `--context`	per indicare un formato contestuale standard;

Opzione	Descrizione
`-u` `--unified`	per indicare un formato contestuale unificato.

22.12.2.3 Differenze multiple e directory

Un file di differenze che contiene informazioni su più coppie di file, deve essere di tipo contestuale (standard o unificato). Quando è stato generato facendo riferimento al contenuto di una directory, i nomi dei file presi in considerazione contengono l'indicazione di un percorso; pertanto, per riprodurre le modifiche in ambito locale, occorre tenere conto della posizione in cui cominciano a trovarsi i dati.

Inoltre, la directory corrente, nel momento in cui si avvia il programma '**patch**', è importante per determinare quali siano i file a cui si devono applicare le modifiche.

Tabella 22.70. Opzioni relative alla posizione in cui si esegue l'applicazione delle modifiche.

Opzione	Descrizione
`-d` ***directory_di_riferimento*** `--directory=`***directory_di_riferimento***	Questa opzione permette di definire la directory di lavoro per '**patch**'.
`-p`***n*** `--strip=`***n***	In questo modo è possibile «togliere» un numero stabilito di barre oblique di separazione all'interno dei percorsi indicati per i file a cui applicare le modifiche. Questa opzione è praticamente obbligatoria in presenza di file di differenze in cui le informazioni sui file contengono un percorso. In generale, quando queste vengono applicate in un contesto equivalente a quello nel quale sono state generate, si utilizza l'opzione '**-p0**', che indica il mantenimento della situazione attuale.

Segue la descrizione di alcuni esempi.

- `$ patch -d ~/prove < prova.diff` [*Invio*]

  Prima di applicare le modifiche contenute nel file di differenze '`prova.diff`', si sposta nella directory '`~/prove/`'.

- `$ patch -p0 < prova.diff` [*Invio*]

  Applica le modifiche contenute nel file '`prova.diff`' che presumibilmente contiene informazioni sui percorsi. L'opzione '**-p0**' garantisce che a partire dalla directory corrente si articolano gli stessi percorsi che appaiono nel file di differenze.

- `$ patch -p1 < prova.diff` [*Invio*]

  Applica le modifiche contenute nel file '`prova.diff`' che presumibilmente contiene informazioni sui percorsi. L'opzione '**-p1**' richiede l'eliminazione della prima barra obliqua nei percorsi, con l'intento presumibile di eliminare il primo livello di directory. Se all'interno del file di differenze si fa riferimento al file '`x/y/z/prova`', significa che le modifiche relative vanno applicate localmente al file '`y/z/prova`'.

Nel caso in cui all'interno del file di differenze si facesse riferimento al file '`./x/y/z/prova`', eliminando la prima barra obliqua di questo percorso, non si otterrebbe alcun cambiamento, dal momento che ciò produrrebbe il percorso '`x/y/z/prova`' che è equivalente al primo. Questo significa che prima di decidere quante barre oblique togliere da un percorso, occorre osservare il contenuto del file di differenze.

In modo analogo, nel caso in cui all'interno del file di differenze si facesse riferimento al file '`/x/y/z/prova`' che, come si vede, è indicato con un percorso assoluto a partire dalla radice, eliminando la prima barra obliqua si ottiene un percorso relativo: '`x/y/z/prova`'.

- `$ cd ~/linux` [*Invio*]

  ```
  $ gzip -d ↩
  ↪     -c ../usb-2.4.0-test2-pre2-for-2.2.16-v3.diff.gz ↩
  ↪| patch -p1 [Invio]
  ```

  Questo è un esempio più complesso ma comune. Si tratta dell'applicazione del file di differenze '`usb-2.4.0-test2-pre2-for-2.2.16-v3.diff.gz`', che come si nota è anche compresso, contenuto nella directory personale dell'utente che compie l'operazione. Nell'esempio si intende che si tratti di modifiche relative ai sorgenti di un kernel Linux collocato a partire dalla directory '`~/linux/`'. In questo caso, il file di differenze inizia in questo modo:

  ```
  --- linux-2.2.16/Documentation/Configure.help   Mon Jun 19 11:26:22 2000
  +++ linux/Documentation/Configure.help  Mon Jun 19 12:02:12 2000
  ```

  Pertanto, essendo già la directory corrente corrispondente a '`linux/`', l'opzione '**-p1**' risolve tutti i problemi.

22.12.2.4 Conservazione delle versioni precedenti

In condizioni normali, '**patch**' sovrascrive i file a cui si applicano le modifiche. Per evitarlo è possibile definire precisamente il nome del file da generare, oppure si può gestire il sistema di mantenimento delle versioni precedenti, utilizzando in particolare l'opzione '**-b**'.

Tabella 22.72. Opzioni relative al controllo delle versioni precedenti.

Opzione	Descrizione
`-o` ***file_aggiornato*** `--output=`***file_aggiornato***	Invece di modificare il file originale, ne crea uno nuovo, utilizzato il nome indicato come argomento dell'opzione.
`-b` `--backup`	Attiva la conservazione della versioni precedenti. In condizioni normali, con questa opzione si ottiene di salvare i file, prima del loro aggiornamento, utilizzando l'estensione aggiuntiva '`.orig`'.
`-z` ***suffisso_di_backup*** `--suffix=`***suffisso_di_backup***	Permette di definire il suffisso (ovvero l'estensione) da utilizzare per le eventuali copie di sicurezza delle versioni precedenti. Se non viene specificato con questa opzione, si utilizza il simbolo contenuto nella variabile di ambiente ***SIMPLE_BACKUP_SUFFIX***. Se anche questa variabile non è stata predisposta, si utilizza l'estensione '`.orig`'.
`-V` ***tipo_di_backup*** `--version-control=`***tipo_di_backup***	Permette di definire esplicitamente il modo con cui gestire le copie di sicurezza delle versioni precedenti, quando si usa anche l'opzione '**-b**'.
`-V` {`t`|`numbered`} `--version-control=`{`t`|`numbered`}	Fa in modo che le copie di sicurezza abbiano un'estensione numerata.

Opzione	Descrizione
`-V {nil|existing}` `--version-control={nil|existing}`	Fa in modo che vengano conservate le copie di sicurezza solo per i file che hanno già una o più copie di sicurezza numerate.
`-V {never|simple}` `--version-control={never|simple}`	Fa in modo che venga conservata una sola copia di sicurezza.

Tabella 22.73. Variabili di ambiente relative al controllo delle versioni precedenti.

Variabile	Descrizione
`PATCH_VERSION_CONTROL`	Permette di definire la modalità di gestione delle copie di sicurezza delle versioni precedenti in modo predefinito. I valori attribuibili a questa variabile sono gli stessi utilizzati come argomento dell'opzione '**`-V`**'.
`VERSION_CONTROL`	Questa variabile ha lo stesso significato di ***PATCH_VERSION_CONTROL***, ma viene presa in considerazione solo in mancanza di questa.
`SIMPLE_BACKUP_SUFFIX`	Definisce il simbolo da utilizzare come suffisso per i nomi dei file che rappresentano le copie di sicurezza.

Segue la descrizione di alcuni esempi.

- `$` **`patch -o aggiornato < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`' generando il file '`aggiornato`', senza toccare i file originali.

- `$` **`patch -b < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`', avendo cura di fare una copia di sicurezza dei file che aggiorna, prima di modificarli.

- `$` **`patch -b -z .vecchio < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`', avendo cura di fare una copia di sicurezza dei file che aggiorna utilizzando per questo l'estensione '`.vecchio`', prima di modificarli.

- `$` **`patch -b -V numbered < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`', avendo cura di fare una copia di sicurezza dei file che aggiorna utilizzando per questo un'estensione contenente un numero progressivo, prima di modificarli. La prima di queste copie di sicurezza ottiene l'estensione '`.~1~`', la seconda '`.~2~`' e così di seguito.

22.12.2.5 Applicazione di modifiche imperfette

'**`patch`**' è generalmente in grado di applicare delle modifiche anche a file che non sono perfettamente identici a quelli con cui sono stati costruiti i file di differenze. Tuttavia, ci sono situazioni in cui '**`patch`**', da solo, non è in grado di poter prendere una decisione autonoma.

Può capitare che i file di modifiche vengano alterati involontariamente, per esempio a causa di una trasmissione attraverso la posta elettronica o per una modifica attraverso un programma per la gestione di file di testo. In questi casi potrebbero essere alterate le spaziature orizzontali attraverso una sostituzione dei caratteri di tabulazione con caratteri spazio, o viceversa. Un problema del genere può essere risolto utilizzando l'opzione '**`-l`**'.

`-l` | `--ignore-white-space`

In questo modo, una sequenza di spazi qualunque equivale a un'altra sequenza di spazi, indipendentemente dal fatto che siano stati usati caratteri di tabulazione, o caratteri spazio veri e propri, ma anche indipendentemente dalla loro quantità.

22.12.2.6 Altre anomalie

Quando '**`patch`**' incontra dei problemi che non è in grado di risolvere da solo, richiede un intervento, ponendo delle domande all'utente. Se ciò accade, si può decidere di guidare '**`patch`**' nell'applicazione delle modifiche o di interrompere il procedimento.

Tutte le modifiche rigettate, vengono salvate in file terminanti con l'estensione '`.rej`', a meno che sia stabilito diversamente con l'opzione '**`-r`**'.

`-r` ***file_degli_errori*** | `--reject-file=`***file_degli_errori***

Con questa opzione, in pratica, si stabilisce direttamente il nome del file che deve contenere le informazioni sulle modifiche che non sono state applicate per qualunque motivo.

22.12.2.7 Differenze invertite

Alla fine delle sezioni dedicate alla creazione di un file di differenze è stato chiarito che l'ordine in cui vanno indicati i file o le directory da confrontare, deve essere tale da avere prima l'oggetto che rappresenta la versione precedente e dopo quello che rappresenta la versione aggiornata.

Alle volte si hanno per le mani file di differenze ottenuti in modo inverso rispetto alle intenzioni reali, pertanto occorre richiedere a '**`patch`**' di adeguarvisi, se possibile.

Tabella 22.74. Opzioni relative all'ordine con cui applicare gli aggiornamenti.

Opzione	Descrizione
`-R` `--reverse`	Richiede a '**`patch`**' di intendere il file di differenze in modo inverso rispetto a quello che sembrerebbe.
`-N` `--forward`	Richiede esplicitamente di ignorare le modifiche che sembrano essere state invertite, oppure che sembrano essere già state applicate.

22.13 Elaborazioni matematiche

Benché esistano dei programmi specifici per le elaborazioni matematiche, come BC (sezione 22.14), alle volte dei programmi di servizio più semplici possono essere ugualmente utili.

22.13.1 Utilizzo di «factor»

Il programma di servizio '**`factor`**'[39] consente di calcolare i fattori di un numero, ovvero la serie di numeri primi, moltiplicando i quali si ottiene il numero di partenza.

`factor` [***numero***]...

Si osservi l'esempio seguente:

```
$ factor 12345 [Invio]

12345: 3 5 823
```

Il comando restituisce un messaggio con il quale si comprende che si ottiene 12345 moltiplicando i numeri primi 3, 5 e 823.

22.13.2 Utilizzo di «seq»

Il programma di servizio '**`seq`**'[40] consente di ottenere una sequenza numerica a partire da un valore iniziale, fino a un valore finale massimo, con una cadenza prestabilita:

```
seq [opzioni] [numero_iniziale [passo]] numero_finale
```

Se si omette l'indicazione del passo, si intende una scansione unitaria; inoltre, se si omette anche il valore iniziale, si intende implicitamente che si tratti di uno. In pratica, l'esempio seguente restituisce attraverso lo standard output la sequenza dei numeri interi da uno a quattro:

```
$ seq 1 1 4 [Invio]

1
2
3
4
```

La stessa cosa si ottiene con i due esempi seguenti, perché il passo e il valore iniziale sono quelli predefiniti:

```
$ seq 1 4 [Invio]
```

```
$ seq 4 [Invio]
```

Attraverso le opzioni è possibile controllare il formato dell'elenco di numeri, così come si può cambiare anche il modo di separarli. Tuttavia, rimane il fatto che i valori ottenuti sono espressi in base 10; per ottenere dei valori con basi di numerazione differenti, occorre un'elaborazione successiva con altri programmi.

Tabella 22.77. Alcune opzioni.

Opzione	Descrizione
`-f` ***formato*** `--format=`***formato***	Definisce esplicitamente il formato di rappresentazione dei valori restituiti. Il formato si esprime attraverso una stringa tra quelle elencate di seguito: '**%g**', notazione normale per i numeri piccoli e scientifica per i valori molto grandi; '**%f**', notazione normale con l'indicazione dei decimali; '**%e**', notazione scientifica.
`-s` ***stringa_di_separazione*** `--separator=`***stringa_di_separazione***	Definisce una stringa di separazione differente dal codice di interruzione di riga.
`-w` `--equal-width`	Richiede espressamente di rappresentare i valori con lo stesso numero di cifre, con l'aggiunta eventuale di zeri anteriori.

Segue la descrizione di alcuni esempi. Si osservi che in questi si mostra l'uso della virgola per separare la parte intera del numero, in base alla localizzazione italiana.

- `$ seq 9 0,5 11` [Invio]

```
9
9,5
10
10,5
11
```

- `$ seq -f %e 9 0,5 11` [Invio]

```
9,000000e+00
9,500000e+00
1,000000e+01
1,050000e+01
1,100000e+01
```

- `$ seq -w 9 0,5 11` [Invio]

```
09,0
09,5
10,0
10,5
11,0
```

- `$ seq -s ' ' -w 9 0,5 11` [Invio]

```
09,0 09,5 10,0 10,5 11,0
```


Come già accennato, per ottenere una sequenza espressa in una base di numerazione diversa, occorre l'aiuto di altri programmi, come negli esempi successivi, dove si usa '**printf**' per rielaborare il risultato emesso da '**seq**' (il primo esempio serve a mostrare cosa si otterrebbe direttamente da '**seq**').

- `$ seq 9 2 13` [Invio]

```
9
11
13
```

- `$ printf %x'\n' 'seq 9 2 13'` [Invio]

```
9
b
d
```

- `$ printf %o'\n' 'seq 9 2 13'` [Invio]

```
11
13
15
```

22.14 BC: linguaggio aritmetico a precisione arbitraria

BC, ovvero *Basic calculator*, è un interprete di un linguaggio aritmetico, che fa parte della tradizione dei sistemi Unix, tanto da essere codificato anche nello standard POSIX. Come linguaggio non ha nulla di speciale, ma la sua facilità di utilizzo in modo interattivo e la sua diffusione, lo rendono molto comodo e utile.

L'utilizzo più conveniente di BC è probabilmente quello a riga di comando, come calcolatrice, tenendo conto che questa sua caratteristica può anche essere sfruttata utilmente all'interno di script di shell. L'esempio seguente mostra un utilizzo interattivo, per comprendere di cosa si tratta, almeno a prima vista:

```
$ bc [Invio]

255*63*3737*512 [Invio]

  30737871360

[Ctrl d]
```

Quello che si vede nell'esempio è la moltiplicazione di tre numeri: 255, 63, 3737 e 512. Il risultato è ciò che si vede alla fine: 30737871360. La stessa cosa si potrebbe inserire in uno script di shell nel modo seguente, in cui il risultato della moltiplicazione viene assegnato alla variabile ***RISULTATO***:

```
…
RISULTATO='echo "255*63*3737*512" | bc'
…
```

Tuttavia, BC è in realtà un linguaggio di programmazione, benché semplice, la cui caratteristica fondamentale è quella di poter definire l'approssimazione del risultato, indipendentemente dall'architettura dell'elaboratore per il quale è stato compilato.

22.14.1 Base di numerazione

Una caratteristica importante di BC è la possibilità di gestire basi di numerazione diverse da 10. Tuttavia, ciò può creare degli imprevisti inattesi, per cui occorre fare attenzione quando si tenta di modificare la convenzione normale.

La base di numerazione viene modificata intervenendo attraverso due variabili predefinite che fanno parte del linguaggio, denominate ***ibase*** e ***obase***. La prima contiene la base di numerazione per i numeri che vengono inseriti, mentre la seconda contiene la base usata per la rappresentazione dei risultati. In condizioni normali, sia ***ibase***, sia ***obase***, contengono il valore 10. Tuttavia, quando si cambia il valore di ***ibase*** si possono creare delle complicazioni; supponendo di voler inserire valori in base otto, basta agire come segue:

```
ibase=8 [Invio]
```

Nel momento in cui si scrive un valore, questo viene interpretato in base otto:

777 [*Invio*]

```
511
```

Infatti, 777_8 equivale a 511.

Quando però si vuole intervenire nuovamente sulla variabile ***ibase***, occorre ricordare che per il momento la base di numerazione è otto. Pertanto, volendo tornare alla base 10, bisogna trasformare prima il valore in ottale: 12_8.

ibase=12 [*Invio*]

777 [*Invio*]

```
777
```

Diversamente, scrivendo nuovamente '**ibase=10**' non si cambierebbe la base di numerazione, perché quel numero andrebbe inteso in ottale.

Esiste anche una convenzione, per cui i valori numerici espressi con una sola cifra, vanno intesi correttamente, in modo indipendente dal valore della variabile ***ibase***. Pertanto, '**9**' vale sempre come se fosse scritto in base dieci, dal momento che non ci possono essere ambiguità anche se la base di numerazione fosse più grande.

Le cifre che possono essere usate per comporre un numero sono i simboli da '**0**' a '**9**', con le lettere maiuscole da '**A**' a '**F**'. In questo modo si possono rappresentare agevolmente numeri con basi di numerazione che vadano da 2 a 16, mentre per basi di numerazione superiori le cose si complicano. In pratica, si possono rappresentare basi superiori scrivendo il risultato a cifre separate, dove ogni cifra è espressa come un numero in base dieci. Per esempio, la stringa «100», esprimente un numero in base venti, verrebbe rappresentata come '**01 00 00**'.[41] L'esempio seguente mostra in che modo arrivare a questo risultato, tenendo in considerazione il fatto che la variabile ***ibase*** contenga inizialmente il valore 10.

obase=20 [*Invio*]

400 [*Invio*]

```
01 00 00
```

In base al principio per il quale una cifra numerica singola viene interpretata in modo non ambiguo, indipendentemente dalla base di numerazione stabilita in ingresso con la variabile ***ibase***, si può tornare facilmente a un inserimento di valori in base 10, sfruttando la cifra '**A**', il cui valore è sempre pari a 10:

ibase=A [*Invio*]

22.14.2 Approssimazione

La variabile ***scale*** definisce la quantità di cifre decimali da prendere in considerazione, quando il contesto non esprime già questo valore. In altri termini, una moltiplicazione definisce già la quantità di cifre decimali da considerare:

10*2.45 [*Invio*]

```
24.50
```

Al contrario, nel caso della divisione è necessario stabilire subito la quantità di decimali da considerare:

scale=4 [*Invio*]

10/3 [*Invio*]

```
3.3333
```

Generalmente, all'avvio dell'interprete, il valore della variabile ***scale*** è pari a zero, avendo così un'approssimazione predefinita alla parte intera.

22.14.3 Linguaggio di programmazione

Il linguaggio di BC ha una vaga somiglianza con il C. In generale, le righe vuote e quelle bianche vengono ignorate, così come il testo circoscritto tra '**/***' e '***/**' (proprio come nel C).

Alcune realizzazioni di BC prevedono anche l'uso del simbolo '**#**' come commento, allo scopo di poter realizzare facilmente degli script, iniziando con '**#!/usr/bin/bc**', ma si tratta di un'estensione che non fa parte dello standard POSIX.

Le istruzioni del linguaggio BC terminano normalmente alla fine della riga, ma è possibile usare anche il punto e virgola ('**;**') se si preferisce, oppure se si vogliono indicare più istruzioni assieme sulla stessa riga. La continuazione di un'istruzione in una riga successiva si ottiene mettendo una barra obliqua inversa ('\') alla fine, esattamente prima del codice di interruzione di riga.

istruzione [**;**]

istruzione \
continuazione_istruzione

Si possono definire dei raggruppamenti di istruzioni, racchiudendoli tra parentesi graffe ('**{ }**'). Anche in questo caso le istruzioni possono essere separate attraverso interruzioni di riga, oppure con il punto e virgola.

{***istruzione***
istruzione
istruzione}

{***istruzione*** ; ***istruzione*** ; ***istruzione***}

Il linguaggio consente la dichiarazione di variabili e di funzioni, che possono avere un nome composto esclusivamente da una lettera minuscola. Alcune realizzazioni di BC consentono l'uso di nomi più articolati, ma si tratta di estensioni non compatibili con le specifiche POSIX.

Il linguaggio BC non prevede una funzione principale, come avviene invece in C. Infatti, si tratta di un linguaggio interpretato dove tutto viene eseguito appena possibile; anche le funzioni esistono appena dichiarate e possono essere sostituite da una dichiarazione successiva che utilizza lo stesso nome

Esistono solo due tipi di dati: le stringhe delimitate e i valori numerici (con la quantità stabilita di cifre dopo la virgola), dove la separazione tra parte intera e parte decimale si indica esclusivamente con un punto ('**.**'). Tuttavia, alle variabili si possono assegnare solo numeri, così come le funzioni possono restituire solo valori numerici.

22.14.3.1 Variabili semplici e array

La dichiarazione di una variabile avviene in modo molto semplice, con l'assegnamento di un valore numerico, come nell'esempio seguente:

```
x=123.456
```

Nello stesso modo si possono dichiarare degli array a una sola dimensione, indicando un indice tra parentesi quadre, come nell'esempio seguente, dove in particolare l'indice è espresso da un'espressione:

```
x[1+2]=234.567
```

Gli array non devono essere dimensionati e possono usare la quantità massima di elementi disponibili in base alla realizzazione di BC.

Il primo elemento si raggiunge con l'indice zero e gli elementi successivi sono numeri interi positivi. Se si fa riferimento a un elemento dell'array che non è ancora stato assegnato, si ottiene il valore zero.

Per fare riferimento a un array nel suo complesso, si indica il nome, seguito dalle parentesi quadre, aperte e chiuse, senza contenere alcun indice: '***x*`[]`**'.

22.14.3.2 Funzioni

La dichiarazione di una funzione ha una forma precisa, dove in questo caso ***x*** rappresenta il nome della stessa:

```
define x([parametro[, parametro]]…) {
    [auto variabile_automatica[, variabile_automatica]…]
    [istruzione]
    …
    …
    [return [(varlore_restituito)]]
}
```

Si osservi in particolare l'uso delle parentesi graffe per delimitare il corpo della funzione: è indispensabile che la parentesi graffa aperta si trovi sulla stessa riga iniziale della dichiarazione della funzione, con i parametri relativi.

I parametri della funzione possono essere nomi di variabili normali, oppure nomi di array senza un indice tra le parentesi quadre.

I parametri che appaiono tra parentesi tonde, equivalgono alla dichiarazione implicita di variabili locali, definite di tipo ***automatico***, contenenti il valore trasmesso al momento della chiamata.

Oltre alle variabili che compongono l'elenco dei parametri della funzione, si possono dichiarare altre variabili automatiche nel modo seguente, nella riga immediatamente successiva alla parentesi graffa aperta:

```
[auto variabile_automatica[, variabile_automatica]…]
```

Si può usare una sola istruzione '**`auto`**', nella quale vanno elencate tutte le variabili automatiche, compresi gli array, nella forma '***x*`[]`**'.

Una funzione restituisce sempre un valore numerico, anche se non viene utilizzata esplicitamente l'istruzione '**`return`**'; in tal caso, si tratta sempre di zero.

Se il valore restituito dalla funzione non viene usato nella chiamata per un assegnamento, questo viene visualizzato, anche se ciò non fosse desiderabile. Per evitare questo inconveniente, è possibile assegnare a una variabile fittizia il valore restituito dalla funzione.

Anche se è possibile fornire un array come parametro in una chiamata di funzione, l'istruzione '**`return`**' non può restituire un array.

La chiamata di una funzione avviene nel modo seguente; anche in questo caso ***x*** rappresenta il nome della funzione chiamata:

```
x([parametro[, parametro]]…)
```

I parametri possono essere variabili oppure valori costanti. Nel primo caso, se la funziona cambia il contenuto delle variabili corrispondenti, tali modifiche non si ripercuotono nelle variabili usate nella chiamata.

Le funzioni possono anche non avere parametri; in quei casi si indicano le parentesi tonde senza alcun contenuto, sia nella dichiarazione, sia nella chiamata.

L'esempio seguente, molto semplice, mostra la dichiarazione di una funzione che esegue la moltiplicazione:


```
define m (x, y) {
    auto z
    z=x*y
    return (z)
}
```

Se questa funzione venisse salvata nel file '`moltiplica`', si potrebbe usare BC nel modo seguente:

```
$ bc moltiplica [Invio]

m (7, 2) [Invio]

  14

[Ctrl d]
```

La parola chiave '**`return`**', può essere usata senza l'indicazione del valore da restituire e quindi senza le parentesi tonde. In tal caso viene restituito il valore zero.

22.14.3.3 Emissione delle informazioni

BC prevede poche funzioni predefinite (interne), ma non mette a disposizione una funzione per l'emissione di stringhe. Se necessario, una costante stringa viene visualizzata semplicemente indicandola come un'istruzione, con un piccolo accorgimento.

Un'espressione che si traduce in un numero, porta alla visualizzazione del risultato, seguito da un codice di interruzione di riga; pertanto, l'esempio seguente genera il risultato 15783552: il valore viene visualizzato e il cursore si trova poi collocato sulla riga successiva:

```
4567*3456 [Invio]

  15783552
  _
```

Al contrario, la visualizzazione di una stringa non fa avanzare alla riga successiva, permettendo l'aggiunta di altre stringhe e di un solo valore numerico finale.

```
"ciao " ; "amore " ; "bello!" [Invio]
```

Infatti, l'esempio si traduce nel testo seguente, con il cursore alla destra del punto esclamativo:

```
  ciao amore bello!_
```

Aggiungendo un numero, la visualizzazione sulla riga termina:

```
"Anni: " ; 35 [Invio]

  Anni: 35
  _
```

Il risultato delle espressioni viene visualizzato se questo non viene catturato da un assegnamento a una variabile. Pertanto, l'esempio seguente visualizza il risultato:

```
7*5 [Invio]

  35
```

Invece, l'esempio successivo non visualizza alcunché:

```
a=7*5 [Invio]
```

Tuttavia, è possibile mostrare il risultato di un'espressione il cui risultato viene assegnato a una variabile, racchiudendola all'interno di parentesi tonde, come nell'esempio seguente:

```
(a=7*5) [Invio]

  35
```

22.14.3.4 Espressioni

Gli operatori che intervengono su valori numerici sono elencati nella tabella 22.101. Esiste tuttavia un chiarimento da fare sull'espressione '***op1%op2***', che non si comporta secondo lo standard comune. Infatti, solo quando ***scale*** contiene il valore zero, il risultato è il resto della divisione intera; diversamente, si ottiene il resto della divisione, tolto il risultato ottenuto in base alla quantità di cifre decimali

stabilito dalla variabile ***scale***. Per esempio, se ***scale*** contiene il valore cinque, 10%3 genera il risultato 0,00001. Infatti, potendo gestire cinque cifre decimali, 10%3 dà il risultato 3,33333, per cui, il resto della divisione rimane solo 0,00001:
3,33333*3+0,00001=10.

Tabella 22.101. Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

Operatore e operandi	Descrizione
`++`*op*	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op`++`	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
`--`*op*	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op`--`	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
`+`*op*	Non ha alcun effetto.
`-`*op*	Inverte il segno dell'operando.
op1 `+` *op2*	Somma i due operandi.
op1 `-` *op2*	Sottrae dal primo il secondo operando.
op1 `*` *op2*	Moltiplica i due operandi.
op1 `/` *op2*	Divide il primo operando per il secondo.
op1 `%` *op2*	Modulo: il resto della divisione tra il primo e il secondo operando.
op1 `^` *op2*	Esponente: il primo operando elevato alla potenza del secondo.
x `=` *valore*	Assegna alla variabile il valore alla destra.
`(` *espressione* `)`	Le parentesi tonde richiedono la precedenza nella valutazione dell'espressione.
op1 `+=` *op2*	*op1* `=` *op1* `+` *op2*
op1 `-=` *op2*	*op1* `=` *op1* `-` *op2*
op1 `*=` *op2*	*op1* `=` *op1* `*` *op2*
op1 `/=` *op2*	*op1* `=` *op1* `/` *op2*
op1 `%=` *op2*	*op1* `=` *op1* `%` *op2*

Alcune realizzazioni tradizionali di BC, non più standard secondo POSIX, consentono l'uso di operatori simili al tipo '***op*`=`', descritti nella tabella, ma invertiti nell'ordine: '`=`***op***'. Ciò crea un problema nella valutazione di alcuni tipi di espressione; per esempio, '`a=-1`' può significare l'assegnamento del valore `-1` alla variabile ***a***, oppure l'assegnamento di '`a-1`'. Per evitare ambiguità in queste condizioni, conviene usare le parentesi: '`a=(-1)`'.

Tabella 22.102. Elenco degli operatori di assegnamento superati che qualche realizzazione di BC potrebbe usare ancora.

Operatore e operandi	Equivalenza
op1 `=+` *op2*	*op1* `=` *op1* `+` *op2*
op1 `=-` *op2*	*op1* `=` *op1* `-` *op2*
op1 `=*` *op2*	*op1* `=` *op1* `*` *op2*
op1 `=/` *op2*	*op1* `=` *op1* `/` *op2*
op1 `=%` *op2*	*op1* `=` *op1* `%` *op2*

Gli operatori di confronto determinano la relazione tra due operandi e possono essere utilizzati esclusivamente in alcuni contesti precisi. Vengono elencati gli operatori disponibili nella tabella 22.103.

Tabella 22.103. Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 `==` *op2*	*Vero* se gli operandi si equivalgono.
op1 `!=` *op2*	*Vero* se gli operandi sono differenti.
op1 `<` *op2*	*Vero* se il primo operando è minore del secondo.
op1 `>` *op2*	*Vero* se il primo operando è maggiore del secondo.
op1 `<=` *op2*	*Vero* se il primo operando è minore o uguale al secondo.
op1 `>=` *op2*	*Vero* se il primo operando è maggiore o uguale al secondo.

Lo standard POSIX stabilisce che queste espressioni possono apparire solo come condizione delle istruzioni '`if`', '`while`' e '`for`'; inoltre, è esclusa la possibilità di comporre espressioni più complesse con l'uso di operatori booleani.

22.14.3.5 Funzioni standard

BC predispone poche funzioni standard, che si distinguono in particolare per la lunghezza del loro nome. Queste sono riepilogate in breve nella tabella 22.104.

Tabella 22.104. Funzioni interne.

Funzione	Valore restituito
`length (` *espressione* `)`	Quantità di cifre significative dell'espressione.
`scale (` *espressione* `)`	Quantità di cifre decimali dell'espressione.
`sqrt (` *espressione* `)`	Radice quadrata dell'espressione.

Per quanto riguarda il caso particolare di ***scale()***, si fa riferimento al numero di decimali che genera l'espressione, in base al contesto. Per esempio, se il valore della variabile ***scale*** è zero, qualunque divisione dà soltanto un risultato intero, per cui ***scale()*** restituisce sempre solo zero.

Oltre a queste funzioni, è possibile chiedere a BC di mettere a disposizione alcune funzioni da una libreria standard. Si tratta di quelle elencate nella tabella 22.105.

Tabella 22.105. Funzioni della libreria standard.

Funzione	Valore restituito
`s (` *x* `)`	Seno.
`c (` *x* `)`	Coseno.
`a (` *x* `)`	Arcotangente.
`l (` *x* `)`	Logaritmo naturale.
`e (` *x* `)`	Funzione esponenziale: *e* elevato alla ***x***.
`j (` *n*`,` *x* `)`	Funzione di Bessel.

22.14.3.6 Strutture di controllo di flusso

Il linguaggio BC gestisce le strutture di controllo di flusso principali, anche se con qualche limitazione. È disponibile una struttura condizionale semplificata (senza l'analisi di un'alternativa), il ciclo iterativo e il ciclo enumerativo:

```
if (condizione) istruzione
```

```
while (condizione) istruzione
```

```
for (espressione1; espressione2; espressione3) istruzione
```

Come nel linguaggio C, dal momento che si possono raggruppare le istruzioni in blocchi racchiusi tra parentesi graffe, in pratica si utilizzano queste strutture nel modo seguente:

```
if (condizione) {
    istruzione
    …
}
```

```
while (condizione) {
    istruzione
    …
}
```

```
for (espressione1; espressione2; espressione3) {
    istruzione
    …
}
```

Naturalmente, le tre espressioni tra parentesi del ciclo enumerativo vanno intese nel modo comune. Per esempio, ciò che appare di seguito serve a mostrare 10 «x», attraverso il conteggio di una variabile.

```
for (i = 0; i < 10; i++) {
    "x"
}
```

Nell'ambito dei cicli, è possibile usare l'istruzione '**break**' per interrompere il ciclo con un'uscita forzata.

22.14.4 Utilizzo di BC

L'interprete del linguaggio BC è l'eseguibile '**bc**', che si utilizza secondo la sintassi seguente:

```
bc [-l] [file_bc]…
```

L'interprete legge ed esegue tutti i file indicati come argomento della riga di comando; alla fine, legge lo standard input. L'interprete termina di funzionare quando il flusso dello standard input termina, oppure quando incontra l'istruzione '**quit**'.

In questo modo, un programma che si deve concludere deve contenere l'istruzione '**quit**', oppure deve essere fornito attraverso lo standard input.

L'opzione '**-l**' serve a ottenere da BC la disponibilità delle funzioni di libreria standard, elencate nella tabella 22.105; inoltre, la variabile ***scale*** viene impostata al valore 20, mentre in condizioni normali il suo valore predefinito è zero.

Lo standard POSIX non prevede l'uso del simbolo '**#**' come commento, per cui non è possibile realizzare degli script se non sfruttando delle estensioni di realizzazioni speciali. In pratica, ci possono essere realizzazioni di BC che consentono di scrivere programmi che iniziano in modo simile a quello seguente, eventualmente con l'aggiunta dell'opzione '**-l**', a cui poi si aggiungono i permessi di esecuzione, ma ciò non è possibile se si vogliono scrivere programmi standard (portabili).

```
#!/usr/bin/bc
```

22.14.5 BC nella realizzazione GNU

La realizzazione GNU di BC[42] consente l'uso di diverse estensioni rispetto allo standard POSIX; in particolare completa la struttura di controllo condizionale con l'alternativa '**else**', aggiunge l'istruzione '**print**' per una gestione migliore della visualizzazione di informazioni e consente l'uso di operatori booleani nelle espressioni logiche, che possono essere usate anche al di fuori del contesto restrittivo stabilito da POSIX. Tuttavia è possibile richiedere un funzionamento strettamente aderente allo standard POSIX, utilizzando l'opzione '**-s**', oppure creando la variabile di ambiente ***POSIXLY_CORRECT***.

L'eseguibile '**bc**' consente l'uso di più opzioni della riga di comando, alcune delle quali vengono descritte brevemente nel seguito.

Opzione	Descrizione
`-l` `--mathlib`	Richiede l'uso delle librerie matematiche standard, impostando la variabile ***scale*** al valore 20.
`-w` `--warn`	Segnala l'uso di estensioni allo standard POSIX.
`-s` `--standard`	Restringe il funzionamento allo standard POSIX.

22.15 Creazione e modifica di file di testo: VI

Il programma più importante che è necessario conoscere quando ci si avvicina a un nuovo sistema operativo è quello che permette di creare e modificare i file di testo normale. Nel caso dei sistemi Unix, è indispensabile conoscere VI (*Visual*), oltre ad altri applicativi simili che possono risultare più comodi da usare. VI[43] [44] [45] [46] è più importante perché onnipresente, ma soprattutto è previsto e richiesto dallo standard POSIX, anche se non si tratta di un programma comodo da utilizzare. VI ha una logica di funzionamento tutta sua che ne impedisce l'utilizzo a chi non abbia letto qualcosa al riguardo. L'intento è quello di chiarire questa logica, almeno in parte, in modo da facilitarne l'utilizzo in caso di necessità.

Per GNU/Linux, non esiste in circolazione una versione originale di VI, ma tante interpretazioni di questo, con potenzialità più o meno ampliate. Per tale motivo, '`/bin/vi`' è solitamente un collegamento (simbolico o meno) al programma che si utilizza effettivamente. La realizzazione a cui si fa riferimento è nVI, che dovrebbe essere conforme allo standard POSIX 1003.2.

La pronuncia corretta del nome sembra essere «v-i-a-i», ovvero, come pronunciare *vee eye* in inglese.

22.15.1 Avvio

L'eseguibile '**vi**' può essere avviato o meno con l'indicazione di un file sul quale intervenire. Se questo file esiste, viene aperto e si ottiene la visualizzazione del suo contenuto per permetterne la modifica. Se non esiste, viene creato.

```
vi [opzioni] [file…]
```

È anche possibile l'indicazione di alcune opzioni, tra cui, le più importanti sono elencate nella tabella successiva.

Tabella 22.109. Alcune opzioni.

Opzione	Descrizione
`-R`	I file vengono aperti (inizialmente) in sola lettura.
`-c` ***comando***	Subito dopo aver caricato il primo dei file degli argomenti, viene eseguito il comando indicato.
`-s` ***file_di_comandi***	Subito dopo aver caricato il primo dei file degli argomenti, vengono eseguiti i comandi contenuti nel file di comandi indicato.

Quando si avvia VI senza indicare alcun file, appare una scherma-

ta simile a quella della figura 22.110 in cui le righe dello schermo contrassegnate dal simbolo tilde (‘**~**’) rappresentano lo spazio non utilizzato dal file.

Figura 22.110. Avvio di VI.

```
_
~
~
~
~
~
~
~
~
/tmp/vi.qbtyNe: new file: line 1
```

22.15.2 Modalità di funzionamento

VI distingue diverse modalità di funzionamento, altrimenti definibili come stati o contesti. Quando si avvia VI, questo si trova di solito nella modalità di comando che permette di usare tasti determinati, attribuendogli significati speciali (di comando). Quando si vuole agire per inserire o modificare del testo, occorre utilizzare un comando con il quale VI passa alla modalità di inserimento e modifica.

Per complicare ulteriormente le cose, c’è da aggiungere che esistono in realtà due tipi di comandi: «visuali» (*visual*) e «due punti» (*colon*). I comandi visuali sono i più semplici e si compongono di sequenze di uno o più tasti il cui inserimento non appare in alcuna parte dello schermo e si concludono senza la pressione del tasto [*Invio*]; i comandi «due punti» iniziano tutti con il simbolo ‘**:**’ (da cui il nome), terminano con [*Invio*] e durante la digitazione appaiono sulla riga inferiore dello schermo. In particolare, questi ultimi sono quelli derivati da EX.

La modalità di inserimento si riferisce al momento in cui è possibile modificare il testo. Per passare dalla modalità di comando a quella di inserimento, si preme il tasto corrispondente alla lettera «i» (inserimento prima del cursore) o alla lettera «a» (inserimento dopo il cursore).

Per tornare alla modalità di comando, da quella di inserimento, è sufficiente premere il tasto [*Esc*]. Quando ci si trova già nella modalità di comando, la pressione del tasto [*Esc*] non produce alcunché o al massimo interrompe l’introduzione di un comando, di conseguenza, se lo si usa inavvertitamente o troppo, non ne derivano inconvenienti.

Lo svantaggio principale di questo tipo di approccio è quello di dover passare alla modalità di comando per qualunque operazione diversa dal puro inserimento di testo. Anche lo spostamento del cursore avviene attraverso dei comandi, obbligando l’utente a premere il tasto [*Esc*] prima di poter utilizzare i tasti per il suo spostamento.

Tuttavia, le realizzazioni più diffuse di VI addolciscono un po’ il suo funzionamento introducendo l’uso dei tasti freccia nel modo consueto dei programmi di scrittura più recenti.

22.15.3 Posizione attiva

Per la descrizione del funzionamento di VI è importante definire il concetto di ***posizione attiva*** che si riferisce al punto in cui si trova il cursore. Estendendo il significato, si può parlare di riga attiva, colonna attiva e parola attiva, intendendo quelle su cui si trova il cursore.

22.15.4 Moltiplicatori

Prima di affrontare i comandi di VI è importante comprendere che l’effetto di molti di questi può essere ***moltiplicato*** utilizzando un numero. Il concetto è molto semplice e si richiama alla matematica: ‘**2a**’ = ‘**a+a**’.

22.15.5 Inserimento

Come già accennato, si può inserire o modificare del testo solo quando si passa alla modalità di inserimento attraverso il comando ‘**i**’ (*insert*) oppure ‘**a**’ (*append*). Durante questa fase, tutti i simboli della tastiera servono per inserire del testo. Con il VI standard si può usare:

- [*Invio*] per terminare una riga e passare alla successiva;
- [*Backspace*] per tornare indietro (nella maggior parte dei casi si ottiene anche la cancellazione del testo);
- [*Esc*] per terminare la modalità di inserimento e passare a quella di comando.

Con le realizzazioni di VI più sofisticate, è concesso normalmente l’uso dei tasti freccia e in alcuni casi anche del tasto [*Canc*].

Per tutte le altre operazioni di modifica del testo si deve passare alla modalità di comando.

Figura 22.111. VI durante la fase di inserimento di testo.

```
Il mio primo documento scritto con VI.

Non e‘ facile, ma ne vale ugualmente la pena_
~
~
~
~
~
~
```

I comandi a disposizione per passare alla modalità di inserimento sono molti e non si limitano quindi ai due modi appena descritti. La tabella 22.112 ne elenca alcuni.

Tabella 22.112. Elenco dei comandi utilizzabili per passare alla modalità di inserimento.

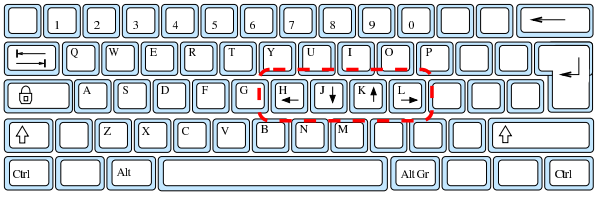
Comando	Descrizione
`I`	Inserisce all’inizio della riga attiva.
`i`	Inserisce prima della posizione attiva.
`A`	Aggiunge alla fine della riga attiva.
`a`	Aggiunge dopo la posizione attiva.
`O`	Inserisce prima della riga attiva (inserendo una riga).
`o`	Aggiunge dopo la riga attiva (inserendo una riga).

22.15.6 Navigazione

Come già accennato, lo spostamento del cursore e di conseguenza della posizione attiva, avviene per mezzo di comandi che generalmente obbligano a terminare la fase di inserimento. Le realizzazioni di VI più recenti permettono l’uso dei tasti freccia durante la modalità di inserimento (oltre che durante la modalità di comando), ma questo è solo un aiuto minimo: in generale è necessario tornare alla modalità di comando.

I comandi normali per lo spostamento del cursore sono le lettere ‘**h**’, ‘**j**’, ‘**k**’ e ‘**l**’, che rispettivamente spostano il cursore a sinistra, in basso, in alto e a destra. La ragione della scelta sta nella vicinanza di queste lettere nella maggior parte delle tastiere.

Figura 22.113. Promemoria visuale per i comandi che permettono lo spostamento del cursore.



Salvo casi particolari e situazioni in cui questo concetto non è ragionevolmente applicabile, i comandi di spostamento, preceduti da un numero, vengono ripetuti tante volte quante ne rappresenta quel

numero. Per esempio, il comando '**2h**' sposta il cursore a sinistra di due posizioni.

Per raggiungere una riga determinata è possibile utilizzare il comando '***n*G**' o '**:*n***' (in questo ultimo caso, seguito da [*Invio*])

Per esempio, per raggiungere la decima riga di un documento ipotetico, si può utilizzare indifferentemente uno dei due comandi seguenti:

```
10G
```

```
:10 [Invio]
```

Per fare scorrere il testo di una schermata alla volta si utilizzano le combinazioni di tasti [*Ctrl B*] e [*Ctrl F*] che rispettivamente spostano il testo all'indietro e in avanti (*back* e *forward*).

I comandi a disposizione per lo spostamento sono ovviamente numerosi, la tabella 22.114 ne elenca alcuni.

Tabella 22.114. Elenco dei comandi utilizzabili per la navigazione all'interno del testo.

Comando	Descrizione
h	Sposta il cursore a sinistra di un carattere.
j	Sposta il cursore in basso nella riga successiva.
k	Sposta il cursore in alto nella riga precedente.
l	Sposta il cursore a destra di un carattere.
-	Sposta il cursore all'inizio della riga precedente.
+	Sposta il cursore all'inizio della riga successiva.
w	Sposta il cursore all'inizio della parola successiva.
e	Sposta il cursore alla fine della parola successiva.
b	Sposta il cursore all'inizio della parola precedente.
^	Sposta il cursore all'inizio della prima parola della riga.
0	Sposta il cursore all'inizio della riga.
$	Sposta il cursore alla fine della riga.
H	Sposta il cursore sulla prima riga che appare sullo schermo.
M	Sposta il cursore sulla riga centrale dello schermo.
L	Sposta il cursore sull'ultima riga che appare sullo schermo.
G	Sposta il cursore sull'ultima riga del file.
***n*G**	Sposta il cursore sulla riga identificata dal numero ***n***.
|	Sposta il cursore sulla prima colonna (all'inizio della riga).
***n*|**	Sposta il cursore sulla colonna identificata dal numero ***n***.
:*n*	Sposta il cursore sulla riga identificata dal numero ***n***.
[*Ctrl B*]	Fa scorrere il testo all'indietro di una schermata.
[*Ctrl F*]	Fa scorrere il testo in avanti di una schermata.
[*Ctrl U*]	Fa scorrere il testo all'indietro di mezza schermata.
[*Ctrl D*]	Fa scorrere il testo in avanti di mezza schermata.

Segue la descrizione di alcuni esempi.

- **5w**

  Sposta il cursore all'inizio della quinta parola successiva.

- **2b**

  Sposta il cursore all'inizio della seconda parola precedente.

- **5G**

  Sposta il cursore all'inizio della quinta riga.

- **4|**

  Sposta il cursore sulla quarta colonna.

22.15.7 Modificatori

I comandi di spostamento, esclusi quelli che iniziano con i due punti ('**:**') e quelli che si ottengono per combinazione ([*Ctrl* ...]), possono essere utilizzati come ***modificatori*** di altri comandi.

All'interno di VI manca il concetto di ***zona di intervento***. Per definire l'estensione di un comando lo si può far precedere da un moltiplicatore (un numero) e in più, o in alternativa, lo si può fare seguire da un comando di spostamento. Il comando di spostamento viene utilizzato in questo caso per definire una zona che va dalla posizione attiva a quella di destinazione del comando, in modo che su questa zona intervenga il comando precedente.

Tuttavia, per poter applicare questo concetto, è necessario che i comandi da utilizzare in associazione con i modificatori (di spostamento), siano stati previsti per questo. Deve trattarsi cioè di comandi che richiedono questa ulteriore aggiunta.

Come viene mostrato in seguito, il comando '**x**' permette di cancellare quello che appare in corrispondenza del cursore. Quando viene premuto il tasto [*x*] si ottiene subito la cancellazione del carattere, pertanto, a questo genere di comandi non si può far seguire alcun modificatore. Questo tipo di comandi può solo essere preceduto da un moltiplicatore.

Si comporta diversamente il comando '**d**' che invece deve essere seguito da un modificatore e con questo definisce una zona da cancellare. Per esempio, '**dw**' cancella dalla posizione attiva fino all'inizio della prossima parola e '**d$**' cancella dalla posizione attiva fino alla fine della riga.

22.15.8 Cancellazione

Durante la fase di inserimento è possibile cancellare solo il carattere appena scritto utilizzando il tasto [*Backspace*], sempre che la realizzazione di VI a disposizione lo consenta, altrimenti si ottiene solo l'arretramento del cursore. Per qualunque altro tipo di cancellazione occorre passare alla modalità di comando.

I comandi di cancellazione più importanti sono '**x**', '**d**' seguito da un modificatore e '**dd**'. Il primo cancella il carattere che si trova in corrispondenza della posizione attiva, cioè del cursore, il secondo cancella dalla posizione attiva fino all'estensione indicata dal modificatore e il terzo cancella tutta la riga attiva. Con VI non è possibile cancellare il carattere che conclude una riga (il codice di interruzione di riga), di conseguenza, per unire due righe insieme si utilizza il comando '**J**' oppure '**j**' (bisogna provare).

Tabella 22.115. Elenco dei comandi utilizzabili per cancellare.

Comando	Descrizione
x	Cancella il carattere che si trova sulla posizione attiva.
J j	Unisce la riga attiva con quella successiva.
dd	Cancella la riga attiva.
d***mod***	Cancella dalla posizione attiva fino all'estensione indicata dal modificatore.
D	agisce come '**d$**'.

Segue la descrizione di alcuni esempi.

- **5x**

  Ripete cinque volte la cancellazione di un carattere. In pratica, cancella cinque caratteri.

- **2dd**

Ripete due volte la cancellazione di una riga. In pratica, cancella la riga attiva e quella seguente.

- **dw**

  Cancella a partire dalla posizione attiva, fino al raggiungimento della prossima parola.

- **2dw**

  Ripete per due volte il tipo di cancellazione dell'esempio precedente. In pratica cancella fino all'inizio della seconda parola.

- **d2w**

  Cancella a partire dalla posizione attiva, fino al raggiungimento della seconda parola successiva. In pratica, esegue la stessa operazione del comando '**2dw**'.

- **db**

  Cancella a ritroso, dalla posizione corrente fino all'inizio della prima parola che viene incontrata.

- **d$**

  Cancella a partire dalla posizione attiva fino alla fine della riga.

- **d5G**

  Cancella dalla posizione attiva fino all'inizio della riga numero cinque.

22.15.9 Sostituzione

La modifica del testo inserito può avvenire attraverso i comandi di cancellazione già visti, oppure attraverso comandi di sostituzione. Generalmente si tratta di comandi che prima cancellano parte del testo e subito dopo attivano l'inserimento.

I comandi di sostituzione più importanti sono '**c**' seguito da un modificatore e '**cc**'. Il primo sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore e il secondo sostituisce tutta la riga attiva.

A fianco di questi se ne aggiungono un paio che possono essere utili proprio per il fatto che non passano alla modalità di inserimento: '**r*x***' e '**~**'. Il primo sostituisce il carattere in corrispondenza del cursore con quello rappresentato da ***x*** e il secondo inverte le lettere minuscole in maiuscole e viceversa.

Tabella 22.116. Elenco dei comandi di sostituzione e rimpiazzo.

Comando	Descrizione
`c`	Sostituisce dalla posizione attiva alla fine della riga.
`cc`	Sostituisce la riga attiva a partire dall'inizio.
`c`***mod***	Sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore.
`r`***x***	Rimpiazza quanto contenuto nella posizione attiva con ***x***.
`~`	Inverte maiuscole e minuscole.

Segue la descrizione di alcuni esempi.

- **cc**

  Sostituisce la riga attiva.

- **c$**

  Sostituisce a partire dalla posizione attiva fino alla fine della riga.

- **rb**

  Rimpiazza il carattere che si trova nella posizione attiva con la lettera «b».

- **10~**

  Inverte le lettere maiuscole e minuscole a partire dalla posizione attiva, per 10 caratteri.

22.15.10 Copia e spostamento di porzioni di testo

La gestione della copia e dello spostamento di testo attraverso VI è un po' complicata. Per questa attività si utilizzano delle aree temporanee di memoria alle quali si possono accodare diverse parti di testo.

L'operazione con la quale si copia una porzione di testo in un'area temporanea di memoria viene detta *yanking*, ovvero estrazione, giustificando così l'uso della lettera «y» nei comandi che compiono questa funzione.

Le aree temporanee di memoria per lo spostamento o la copia di testo possono essere 27: una per ogni lettera dell'alfabeto e una aggiuntiva senza nome.

Il modo più semplice di gestire questo meccanismo è quello di usare l'area temporanea senza nome. Per copiare una porzione di testo si può utilizzare il comando '**y**' seguito da un modificatore, oppure il comando '**yy**' che invece si riferisce a tutta la riga attiva. Per incollare il testo copiato, dopo aver posizionato il cursore nella posizione di destinazione, si può utilizzare il comando '**p**' oppure '**P**', a seconda che si intenda incollare prima o dopo la posizione del cursore.

Il comandi '**p**' e '**P**' non cancellano il contenuto dell'area temporanea, di conseguenza, se serve si può ripetere l'operazione di inserimento riutilizzando questi comandi.

Se invece di copiare si vuole spostare il testo, al posto dei comandi di estrazione si possono usare quelli di cancellazione, che, anche se non è stato chiarito precedentemente, prima di cancellare il testo fanno una copia nell'area temporanea.

Tabella 22.117. Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso dell'area temporanea di memoria senza nome.

Comando	Descrizione
`yy`	Copia la riga attiva nell'area temporanea.
`y`***mod***	Copia nell'area temporanea il testo fino all'estensione indicata dal modificatore.
`dd`	Trasferisce la riga attiva nell'area temporanea.
`d`***mod***	Trasferisce nell'area temporanea il testo fino all'estensione indicata dal modificatore.
`p`	Incolla dopo la posizione del cursore.
`P`	Incolla prima della posizione del cursore.

Segue la descrizione di alcuni esempi.

- **5yy**

  Copia nell'area temporanea cinque righe a partire da quella attiva.

- **yw**

  Copia nell'area temporanea il testo che parte dalla posizione attiva fino all'inizio della prossima parola.

- **y$**

  Copia nell'area temporanea il testo che parte dalla posizione attiva fino alla fine della riga.

- **3dd**

  Sposta nell'area temporanea tre righe a partire da quella attiva.

- **2P**

  Incolla due copie del testo contenuto nell'area temporanea a partire dalla posizione a sinistra del cursore.

22.15.11 Copia e spostamento con nome

Quando si vogliono utilizzare delle aree temporanee di memoria specifiche, cioè identificate attraverso le lettere dell'alfabeto, si procede

nei modi già visti nel caso dell'uso dell'area temporanea senza nome, con la differenza che i comandi sono preceduti da '**"*x***', dove ***x*** è la lettera che si vuole usare.

Si introduce però una novità importante: è possibile aggiungere del testo a un'area temporanea: basta indicarla attraverso una lettera maiuscola.

Tabella 22.118. Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso delle aree temporanee con nome.

Comando	Descrizione
`"`*x*`yy`	Copia la riga attiva nell'area temporanea ***x***
`"`*x*`y`***mod***	Copia nell'area temporanea ***x*** il testo fino all'indicazione dal modificatore.
`"`*x*`dd`	Trasferisce la riga attiva nell'area temporanea ***x***.
`"`*x*`d`***mod***	Trasferisce nell'area temporanea ***x*** il testo fino all'indicazione dal modificatore.
`"`*x*`p`	Incolla il contenuto dell'area temporanea ***x*** prima del cursore.
`"`*x*`P`	Incolla il contenuto dell'area temporanea ***x*** dopo il cursore.

Segue la descrizione di alcuni esempi.

- **`"adw`**

  Sposta il testo nell'area temporanea '**a**', a partire dalla posizione attiva fino all'inizio della prossima parola.

- **`"a5yy`**

  Copia cinque righe nell'area temporanea '**a**', a partire dalla posizione attiva (inclusa).

- **`"A3yy`**

  Aggiunge tre righe nell'area temporanea '**a**', a partire dalla posizione attiva (inclusa).

- **`"a2P`**

  Incolla due copie del contenuto dell'area temporanea '**a**', a partire dalla posizione precedente a quella su cui si trova il cursore.

22.15.12 Ricerche

Per effettuare delle ricerche all'interno del documento aperto con VI si possono utilizzare le espressioni regolari (sezione 23.1) attraverso due comandi un po' strani: '**/**' e '**?**'. La sintassi per la ricerca in avanti è:

/modello

Per la ricerca a ritroso è la seguente:

?*modello*

Nel momento in cui si preme il tasto della barra obliqua o del punto interrogativo, VI visualizza il comando nella riga inferiore dello schermo permettendone il controllo e la correzione come avviene per i comandi che iniziano con i due punti. Al termine, l'inserimento di questo tipo di comando deve essere concluso con un [*Invio*].

Tabella 22.119. I comandi di ricerca attraverso espressioni regolari.

Comando	Descrizione
/modello	Cerca in avanti una corrispondenza con il modello indicato.
?*modello*	Cerca all'indietro una corrispondenza con il modello indicato.
`n`	Ripete l'ultimo comando '**/**' o '**?**'.
`N`	Ripete l'ultimo comando '**/**' o '**?**' in modo inverso .

Il tipo di espressione regolare che può essere utilizzato con VI è solo quello elementare e valgono in particolare le regole indicate nella tabella 22.120.

Tabella 22.120. Le espressioni regolari che dovrebbero essere disponibili con la maggior parte delle realizzazioni di VI.

Simbolo	Descrizione
`.`	Corrisponde a un carattere qualsiasi.
`\`	Fa perdere il significato speciale che può avere il carattere seguente.
`^`	Corrisponde all'inizio di una riga.
`$`	Corrisponde alla fine di una riga.
`[`***abc***`]`	Corrisponde a un carattere qualsiasi tra quelli tra parentesi quadre.
`[^`***abc***`]`	Corrisponde a un carattere qualsiasi diverso da quelli tra parentesi quadre.
`[`***a*****`-`*****z***`]`	Un carattere qualsiasi nell'intervallo compreso tra ***a*** e ***z***.
`[^`***a*****`-`*****z***`]`	Un carattere qualsiasi diverso dall'intervallo compreso tra ***a*** e ***z***.

Segue la descrizione di alcuni esempi.

- **`/[Ll]inux`** [*Invio*]

  Cerca in avanti tutte le stringhe corrispondenti a «Linux» oppure «linux».

- **`/\.$`** [*Invio*]

  Cerca in avanti il punto finale di una riga (si osservi l'uso della barra obliqua inversa per togliere il significato speciale che ha il punto in un'espressione regolare).

22.15.13 Ricerche e sostituzioni

La ricerca e sostituzione sistematica avviene attraverso un comando particolare che inizia con i due punti. La sua sintassi è la seguente:

:inizio**`,`*****fine*****`s/`*****modello_da_cercare*****`/`*****sostituzione*****`/`**`[g][c]`

L'indicazione ***inizio*** e ***fine*** fa riferimento alle righe su cui intervenire. Si possono indicare dei numeri, oppure dei simboli con funzioni simili. Il simbolo '**$**' può essere usato per indicare l'ultima riga del file. Un punto singolo ('**.**') rappresenta la riga attiva. Il simbolo '**%**' viene invece utilizzato da solo per indicare tutte le righe del file.

Il modello utilizzato per la ricerca viene espresso secondo la forma delle espressioni regolari.

Normalmente, il valore da sostituire al modello cercato è fisso, ma può contenere un riferimento alla stringa trovata, attraverso il simbolo '**&**'.

La direttiva '**g**' specifica che si deve intervenire su tutte le occorrenze della corrispondenza con il modello, altrimenti la sostituzione riguarda solo la prima di queste.

La direttiva '**c**' specifica che ogni sostituzione deve essere confermata espressamente.

Segue la descrizione di alcuni esempi.

- **`:1,$s/pippo/pappa/g`** [*Invio*]

  Sostituisce ogni occorrenza della parola «pippo» con la parola «pappa». La ricerca viene effettuata a partire dalla prima riga fino all'ultima.

- **`:%s/pippo/pappa/g`** [*Invio*]

  Questo è un modo alternativo per eseguire la stessa operazione dell'esempio precedente: il simbolo '**%**' rappresenta da solo tutte le righe del file.

- **`:.,10s/^..//g`** [*Invio*]

Elimina i primi due caratteri (‘**^..**’) da 10 righe a partire da quella attiva.

- **:%s/^..//gc** [*Invio*]

  Esegue la stessa operazione dell’esempio precedente, applicando la sostituzione su tutto il file, richiedendo però conferma per ogni sostituzione.

- **:.,10s/^/xxxx/g** [*Invio*]

  Inserisce la stringa «xxxx» all’inizio di 10 righe a partire da quella attiva.

Figura 22.121. VI: ricerche e sostituzioni.

```
:inizio,fine s/modello_da_cercare/sostituzione/[g][c]
   |   |                    |
   |   |                    '--> «&»   riferimento
   |   |                                alla stringa
   |   |                                trovata
   |   |--> «n»   riga n-esima
   |   |--> «$»   ultima riga
   |   '--> «.»   riga corrente
   |--> «n»   riga n-esima
   '--> «.»   riga corrente
```

Attenzione:

```
:%s/modello_da_cercare/sostituzione/[g][c]
```

equivale a:

```
:1,$s/modello_da_cercare/sostituzione/[g][c]
```

22.15.14 Annullamento dell’ultimo comando

VI permette di annullare l’ultimo comando inserito attraverso il comando ‘**u**’. A seconda della realizzazione di VI utilizzata, richiamando nuovamente il comando ‘**u**’ si riottengono le modifiche annullate precedentemente, oppure si continuano ad annullare gli effetti dei comandi precedenti.

Tabella 22.122. Annullamento di un comando.

Comando	Descrizione
u	Annulla l’ultimo comando.
U	Annulla le modifiche sulla riga attiva.

22.15.15 Caricamento, salvataggio e conclusione

Il file o i file utilizzati per la modifica (compresi quelli che si creano) vengono aperti normalmente attraverso l’indicazione nella riga di comando, al momento dell’avvio dell’eseguibile ‘**vi**’.

Il salvataggio di un file può essere fatto per mezzo del comando ‘**:w**’ (*write*) seguito eventualmente dal nome del file (quando si vuole salvare con un nome diverso oppure quando si sta creando un file nuovo). La conclusione dell’attività di VI si ottiene con il comando ‘**:q**’ (*quit*). I comandi possono essere combinati tra loro, per esempio quando si vuole salvare e concludere l’attività simultaneamente con il comando ‘**:wq**’. Il punto esclamativo (‘**!**’) può essere usato alla fine di questi comandi per forzare le situazioni, come quando si vuole concludere l’attività senza salvare con il comando ‘**:q!**’.

Dal momento che VI non mostra normalmente alcuna informazione riferita al file su cui si opera (compreso il nome), il comando ‘**:f**’ (oppure la combinazione [*Ctrl g*]) mostra sulla riga inferiore dello schermo: il nome del file aperto, le dimensioni e il numero della riga attiva.

Tabella 22.123. I comandi per il caricamento dei file e il loro salvataggio.

Comando	Descrizione
:e *nome_file*	Carica il file indicato per poterlo modificare.
:e!	Ricarica il file annullando le modifiche fatte nel frattempo.
:r *nome_file*	Legge il file indicato e ne inserisce il contenuto dopo la riga attiva.
:f	Mostra il nome e le caratteristiche del file aperto.
:w	Salva.
:w *nome_file*	Salva una copia con il nome indicato.
:wq	Salva e termina l’esecuzione.
:q	Fine lavoro.
:q!	Fine lavoro forzato.

Figura 22.124. VI durante l’esecuzione del comando ‘**:w**’, prima della pressione conclusiva del tasto [*Invio*]

```
Il mio primo documento scritto con VI.

Non e' facile, ma ne vale ugualmente la pena
~
~
~
~
~
~
:w esempio_
```

22.15.16 Variabili

VI ha ereditato da EX delle variabili di configurazione. Non si tratta di variabili di ambiente, ma di variabili interne a VI. Per attivare una variabile si utilizza il comando seguente:

```
:set [no]nome_della_variabile
```

Il prefisso ‘**no**’, prima del nome della variabile, serve per disattivarla. La tabella 22.125 mostra l’uso del comando ‘**:set**’ assieme a queste variabili.

Tabella 22.125. VI: modalità di funzionamento.

:set [no]autoindent	Mantiene i livelli di rientro nelle righe nuove.
:set [no]beautify	Elimina i caratteri speciali non stampabili.
:set [no]ignorecase	Nelle ricerche, ignora la differenza tra maiuscole e minuscole.
:set [no]list	Mostra i caratteri di tabulazione e di interruzione di riga.
:set [no]number	Visualizza i numeri delle righe.
:set [no]ruler	Visualizza le coordinate del cursore alla base dello schermo.

Segue la descrizione di alcuni esempi.

- **:set nolist** [*Invio*]

  Disabilita la visualizzazione dei caratteri di tabulazione e di fine riga.

- **:set number** [*Invio*]

  Visualizza i numeri di riga.

22.15.17 Comandi particolari

Nella tabella seguente sono elencati dei comandi particolari che non sono stati inclusi in altre categorie specifiche.

Comando	Descrizione
`m`*x*	Etichetta la posizione corrente con la lettera rappresentata da *x*. Valgono solo le lettere minuscole. Il testo non viene modificato.
`'`*x*	Sposta il cursore all'inizio della riga che contiene l'etichetta rappresentata da *x*.
[*Ctrl L*]	Riscrive la schermata: se sono apparsi messaggi che creano difficoltà alla visualizzazione del testo su cui si sta lavorando, questo comando permette di farli scomparire mostrando il testo effettivo del file.
`:!`*comando*	Esegue il comando di shell indicato.
`:ab` *abbreviazione testo_da_sostituire*	Permette di stabilire un'abbreviazione da sostituire sistematicamente con tutto quello che segue il comando. Se si usa '**`:ab`**' da solo, si ottiene un elenco delle abbreviazioni disponibili.

22.15.18 File di configurazione

Può essere utilizzato il file '`~/.exrc`' per personalizzare la configurazione di VI attraverso comandi *colon* (quelli tipici di EX). Le cose più comuni che possono apparire all'interno di questo file sono la definizione di abbreviazioni e la definizione di alcune variabili. Segue un esempio:

```
:ab lx Linux
:ab xwin X Window System
:set autoindent
:set number
```

22.15.19 Problemi di portabilità

Uno dei vantaggi importanti nell'uso di VI sta nella disponibilità di realizzazioni di questo programma per qualsiasi piattaforma. All'inizio di questo gruppo di sezioni su VI si accennava al fatto che esiste un'ottima versione Dos in grado di funzionare molto bene anche con i vecchi elaboratori dotati di poca memoria.

Quando si trasferiscono testi da un sistema GNU/Linux a Dos e viceversa si pone il problema dell'insieme di caratteri a disposizione: su GNU/Linux si utilizza presumibilmente la codifica UTF-8, oppure Latin 1, mentre con il Dos no.

La soluzione più semplice a questo problema è probabilmente quella di usare Latin 1 in generale e di convertire le lettere accentate Dos in Latin 1 quando possibile. Per questo si può realizzare un semplice file di comandi da eseguire automaticamente utilizzando l'opzione '**`-s`**'.[47]

```
:1,$s/~E/à/g
:1,$s/~J/è/g
:1,$s/~B/é/g
:1,$s/~M/ì/g
:1,$s/~U/ò/g
:1,$s/~W/ù/g
```

In pratica, la sintassi da usare all'avvio di VI dovrebbe essere la seguente:

```
vi -s file_comandi file_da_elaborare
```

In generale, i problemi legati alla conversione di un file da un insieme di caratteri all'altro, si risolvono attraverso l'uso di '**`recode`**', descritto nella sezione 47.8.1.

22.16 File manager: Midnight Commander

Il gestore di file, o *file manager*, è quel tipo di programma che facilita la gestione di file e directory e spesso incorpora anche le funzionalità di una shell. Il programma più importante di questo genere è Midnight Commander, che corrisponde all'eseguibile '**`mc`**'.[48]

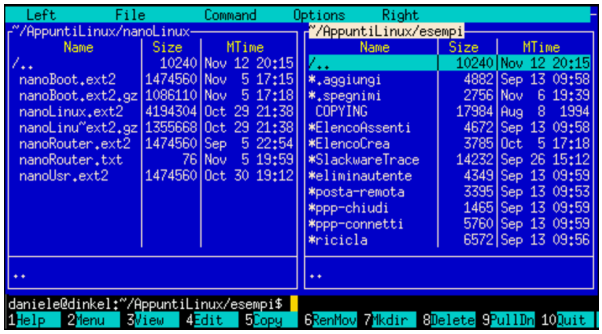
Midnight Commander, è un ottimo gestore di file per i terminali a caratteri. Il nome richiama chiaramente la sua origine: si tratta di un programma molto simile al noto Norton Commander (software proprietario nato per il Dos).

Non si può dire che si tratti di un programma essenziale, dal momento che gli strumenti a disposizione dei sistemi Unix comuni, sono più che sufficienti. Tuttavia, uno strumento del genere può facilitare di molto l'attività dell'utente comune o dell'amministratore del sistema.

22.16.1 Funzionamento

Lo scopo principale di questo programma è quello di permettere la visione e la gestione simultanea di due directory di lavoro.

Figura 22.129. Midnight Commander.



A questo proposito, si può suddividere lo schermo gestito da Midnight Commander in quattro parti:

1. la superficie più grande, nella parte centrale, è utilizzata da due pannelli contenenti l'elenco di due directory;
2. la penultima riga dello schermo viene usata per l'inserimento di comandi di shell;
3. l'ultima riga in basso mostra i riferimenti ai tasti funzionali;
4. la prima riga, in alto, è utilizzata per la barra del menù e può restare eventualmente invisibile fino a che il menù non viene richiamato.

Dei due pannelli, uno è quello attivo e di conseguenza è quella la directory corrente. Il pannello attivo è evidenziato dalla presenza della barra di selezione. La maggior parte delle operazioni hanno effetto sul pannello attivo e quando l'operazione richiede un riferimento a una destinazione, si fa riferimento alla directory dell'altro pannello.

La presenza di una riga di comando permette di inserire ed eseguire comandi di shell nel modo solito. Per terminare il funzionamento di Midnight Commander si usa il tasto [*F10*], oppure, spesso è possibile usare anche il comando '**`exit`**' come si fa con le shell comuni.

22.16.2 Avvio di Midnight Commander

L'eseguibile '**`mc`**' viene avviato normalmente senza alcun argomento. Eventualmente, se si indicano una o due directory, il contenuto di queste viene visualizzato inizialmente:

```
mc [opzioni] [directory_1 [directory_2]]
```

Tabella 22.130. Alcune opzioni.

Opzione	Descrizione
`-b`	Forza Midnight Commander a funzionare in modo monocromatico.
`-d`	Disabilita l'uso del mouse.

22.16.3 Uso del mouse

L'uso del mouse, quando la piattaforma lo consente, è abbastanza intuitivo:

- un clic con il primo tasto (quello sinistro) su un file di una directory, lo seleziona, attivando automaticamente il pannello relativo;
- un clic con il terzo tasto (quello destro) marca o toglie una marcatura a un file;
- un clic doppio su un file provoca
  - la sua esecuzione se si tratta di un eseguibile,
  - l'apertura della directory, se si tratta di una directory,
  - l'esecuzione di un'azione abbinata all'estensione del file, se non si tratta di un eseguibile;
- le funzioni associate ai tasti funzionali possono essere eseguite facendo un clic sull'etichetta di questi posta sull'ultima riga dello schermo;
- le voci del menù possono essere aperte con un clic, ma se queste sono nascoste, basta un clic sulla prima riga dello schermo per farle apparire.

22.16.4 Tastiera

Midnight Commander potrebbe essere utilizzato anche in situazioni non ottimali come lo è invece una console di GNU/Linux. In questi casi, il mouse potrebbe mancare e la tastiera potrebbe non rispondere nel modo consueto, come quando si sta utilizzando una connessione TELNET. Esistono quindi diversi modi per fare le stesse cose e un minimo di nozioni può fare risparmiare molto tempo.

Il programma fornisce dei suggerimenti e una guida interna, in cui viene usata una notazione per le combinazioni di tasti, tipica di Emacs. In pratica, '**`C-a`**' rappresenta la combinazione [*Ctrl a*], mentre '**`M-a`**' rappresenta la combinazione [*Meta a*] ([*Alt a*] negli elaboratori x86).

In particolare, dal momento che non tutte le tastiere hanno un tasto [*Meta*], oppure alcune connessioni TELNET non ne permettono l'uso, in sua sostituzione si può utilizzare la sequenza [*Esc*][*tasto*]. Si tratta quindi di premere [*Esc*], rilasciarlo e premere subito dopo il tasto successivo.[49]

Anche i tasti funzionali possono creare dei problemi. In tal caso si può utilizzare la combinazione [*Meta* ***n***], dove il 10 corrisponde allo zero. Per esempio, al posto del tasto [*F1*] si può utilizzare la combinazione [*Meta 1*], mentre per [*F10*] si può usare [*Meta 0*].

Il tasto [*Invio*] ha diversi utilizzi a seconda del contesto:

- se è stato scritto qualcosa nella riga di comando, viene eseguito il comando;
- se la riga di comando è vuota, si ottiene un'azione equivalente al clic doppio sul file selezionato (quello su cui si trova la barra di selezione);
- se è aperta una finestra di dialogo, esegue l'azione corrispondente al pulsante grafico evidenziato o predefinito.[50]

Il funzionamento di altri tasti e combinazioni di tasti viene riassunto nella tabella 22.131. Quello che è importante tenere presente è che non sono sempre disponibili tutti in ogni situazione.

Tabella 22.131. Alcuni tasti e combinazioni di tasti utili per il controllo di Midnight Commander.

Comando	Alternativa	Descrizione
[*Meta Esc*]	[*Esc*][*Esc*]	Annulla l'ultima operazione o conclude qualcosa.
[*Ctrl l*]		Ridisegna l'immagine dello schermo o della finestra.
[*Ctrl r*]		Rilegge il contenuto della directory aggiornando il pannello attivo.
[*Ctrl s*]	[*Meta s*]	Inizia la ricerca di un file nel pannello attivo.
[*Meta t*]		Passa alla modalità successiva di visualizzazione del pannello attivo.
[+]		Seleziona un gruppo di file, attraverso l'uso di metacaratteri.
[−]	[\]	Deseleziona un gruppo di file, attraverso l'uso di metacaratteri.
[*Tab*]	[*Ctrl i*]	Seleziona l'altro pannello.
[*Ins*]	[*Ctrl t*]	Marca o toglie la marcatura dal file selezionato.
[*freccia-su*]	[*Ctrl p*]	Sposta la barra di selezione in alto di una posizione.
[*freccia-giù*]	[*Ctrl n*]	Sposta la barra di selezione in basso di una posizione.
[*pagina-su*]	[*Meta v*]	Sposta la barra di selezione in alto di una pagina.
[*pagina-giù*]	[*Ctrl v*]	Sposta la barra di selezione in basso di una pagina.
[*Meta Invio*]	[*Esc*][*Invio*]	Copia il nome del file selezionato nella riga di comando.
[*Meta Tab*]	[*Esc*][*Tab*]	Tenta di completare il nome iniziato nella riga di comando.
[*Ctrl q*][***x***]		Inserisce ***x*** nella riga di comando evitando interpretazioni diverse.
[*Meta p*]	[*Esc*][*p*]	Recupera il comando precedente accumulato nello storico dei comandi.
[*Meta n*]	[*Esc*][*n*]	Recupera il comando successivo accumulato nello storico dei comandi.
[*freccia-sinistra*]	[*Ctrl b*]	Sposta il cursore a sinistra in una riga di comando qualunque.
[*freccia-destra*]	[*Ctrl f*]	Sposta il cursore a destra in una riga di comando qualunque.
[*Backspace*]	[*Ctrl h*]	Cancella il carattere precedente.
[*Canc*]	[*Ctrl d*]	Cancella il carattere successivo.

Utilizzando Midnight Commander, occorre fare attenzione all'uso del tasto [*Esc*]. Dal momento che serve a generare delle combinazioni (o meglio delle sequenze), come nel caso in cui si vogliono emulare i tasti funzionali, la semplice pressione di [*Esc*] non genera nulla, fino a che non si preme un altro tasto. Se la sequenza che si genera, [*Esc*][*tasto*], non è conosciuta da Midnight Commander, ciò che si ottiene è l'equivalente di [*Meta Esc*], cioè l'annullamento o la conclusione di qualcosa.

22.16.5 Menù

Il funzionamento della barra del menù è abbastanza intuitivo. In particolare va ricordato che, per attivarlo quando non si dispone del mouse, si usa il tasto [*F9*].

A fianco delle voci contenute nelle tendine dei menù sono annotate le combinazioni della tastiera che possono essere usate come scorciatoia per la funzione relativa.

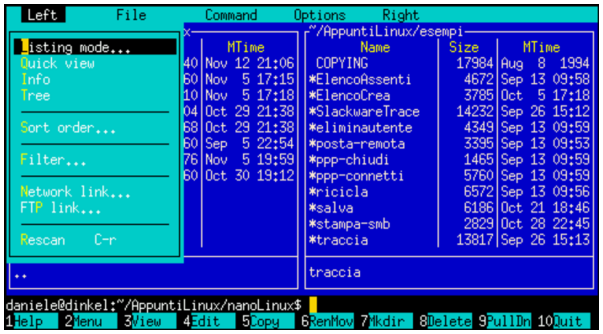
22.16.6 Configurazione

Midnight Commander può essere configurato a vari livelli. Il più semplice è la scelta del modo in cui devono essere usati i pannelli. Per questo si trovano i menù *Left* e *Right*, identici, ma riferiti ai pannelli rispettivi. A livello globale, è possibile definire la configurazione di Midnight Commander attraverso il menù *Options*. In

particolare è da ricordare che occorre salvare la configurazione, se si vuole che sia mantenuta. Il salvataggio di questa genera il file '`~/.mc/ini`'.

La figura 22.132 mostra il menù *Left*, attraverso il quale configurare il comportamento del pannello sinistro. Le stesse voci appaiono nel menù *Right*.

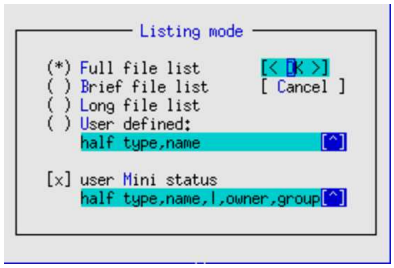
Figura 22.132. Il menù *Left* di Midnight Commander.



- *Listing mode...*
  attiva implicitamente la visualizzazione dell'elenco della directory aperta in quel pannello e ne specifica gli elementi;
- *Quick view*
  visualizza (per quanto possibile) il contenuto del file selezionato nell'altro pannello;
- *Info*
  mostra le informazioni complete del file selezionato nell'altro pannello;
- *Tree*
  mostra la struttura del file system;
- *Sort order*
  permette di definire l'ordinamento dei file che appaiono nel pannello;
- *Filter*
  permette di definire un filtro, espresso attraverso caratteri jolly o espressioni regolari, dei nomi degli elementi contenuti nel pannello.

La funzione *Listing mode* è molto utile per configurare l'aspetto dell'elenco di file del pannello sinistro e destro. La figura 22.133 mostra un esempio della maschera che viene ottenuta attraverso la sua selezione.

Figura 22.133. La maschera a cui si accede attraverso la voce *Listing mode*.



È probabile che sia preferibile attivare la modalità *Full file list* (come si vede nell'esempio), oppure chi è più esperto può trovare utile la possibilità di configurare gli elementi da visualizzare nell'elenco, attraverso la voce *User defined*. In questo caso, occorre compilare il campo successivo, con l'indicazione delle colonne da includere. La prima parola chiave serve a stabilire la dimensione dell'elenco:

- '**`half`**' indica un elenco che utilizza solo metà schermo;
- '**`full`**' indica un elenco che utilizza tutto lo schermo.

Le parole successive rappresentano le colonne, separate da una virgola:

- '**`|`**' rappresenta una riga di divisione tra le colonne;
- '**`type`**' rappresenta un simbolo che permette di identificare il tipo di file;
- '**`name`**' rappresenta il nome del file;
- '**`owner`**' rappresenta il nome dell'utente proprietario;
- '**`group`**' rappresenta il nome del gruppo proprietario;
- '**`size`**' rappresenta la dimensione del file;
- '**`perm`**' rappresenta i permessi del file.

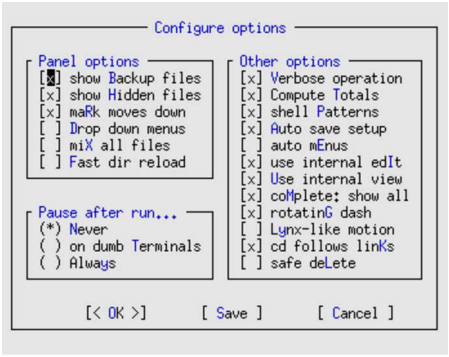
È possibile attivare anche la voce *user Mini status* per visualizzare sul fondo della finestra altre caratteristiche del file evidenziato dal cursore. Gli elementi da visualizzare sono indicati attraverso una sintassi analoga a quanto appena descritto sopra.

Il menù delle opzioni (*Options*) è quello che permette di configurare il funzionamento di Midnight Commander in modo globale, indipendentemente dalle caratteristiche di una finestra particolare. La funzione più importante è appunto *Configuration*, anche se in realtà, tutte le altre voci del menù riguardano la configurazione di questo programma.

La configurazione di Midnight Commander, sia per ciò che riguarda questo menù, sia per quanto accessibile attraverso i menù *Left* e *Right*, viene memorizzata nel file '`~/.mc/ini`'. Per ottenere questo, occorre però che tale configurazione sia salvata attraverso la funzione *Save setup*, alla fine del menù delle opzioni.

La funzione *Configuration* permette di modificare alcuni comportamenti importanti di Midnight Commander. Per esempio è possibile includere o escludere i file «nascosti» e quelli che rappresentano le copie di sicurezza di versioni precedenti. Inoltre, le versioni recenti di Midnight Commander permettono di utilizzare un programma interno per la gestione dei file di testo. Questo, a differenza dei comuni programmi del genere che funzionano su uno schermo a celle di caratteri, si avvicina molto ai programmi simili utilizzati in ambiente Dos. La figura 22.134 mostra un esempio della maschera di configurazione generale delle opzioni.

Figura 22.134. Menù opzioni: configurazione.



22.16.7 File system virtuali

Uno dei vantaggi principali nell'utilizzo di questo programma sta nella facilità con cui è possibile accedere al contenuto di file compressi o a dei servizi FTP. È sufficiente premere [*Invio*] su un archivio, anche compresso, per accedere al suo contenuto (in sola lettura). Anche i pacchetti RPM (Red Hat) e Debian sono accessibili facilmente, purché siano presenti i programmi '**`rpm`**' e '**`dpkg`**'.

Figura 22.135. Midnight Commander permette di accedere facilmente anche al contenuto di archivi compressi.

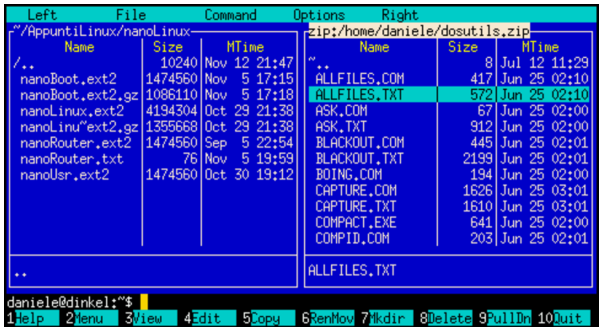


Figura 22.136. Gli archivi RPM vengono gestiti come archivi compressi.

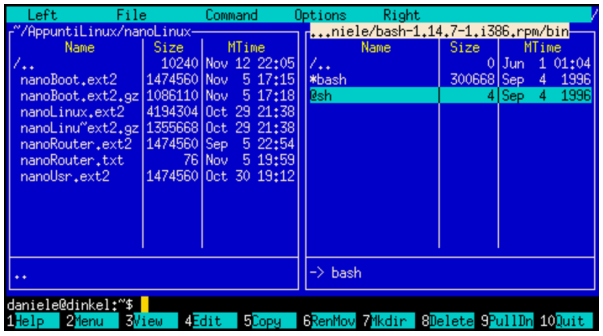
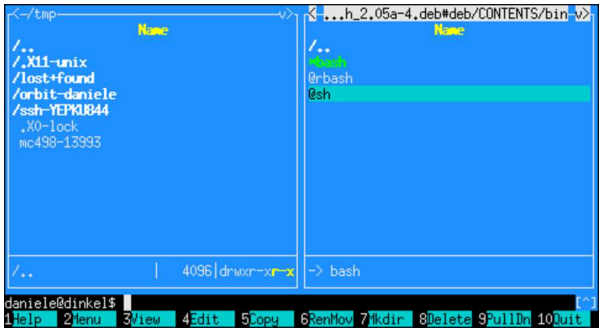


Figura 22.137. Gli archivi Debian vengono gestiti come archivi compressi.



Nello stesso modo è possibile accedere a un servizio FTP, come se si trattasse di un file system normale.

Queste funzionalità sono disponibili tramite il menù, ma si può usare una forma particolare nella riga di comando. Nel caso di FTP, si usa la forma seguente:

```
ftp://[utente@]nodo[directory_remota]
```

Per esempio, per accedere alla directory '`/pub/`' del servizio FTP anonimo presso *dinkel.brot.dg*, si può utilizzare il comando seguente:

```
$ cd ftp://dinkel.brot.dg/pub [Invio]
```

Midnight Commander, così come molti altri programmi clienti FTP, consente anche di inserire il nominativo-utente assieme alla parola d'ordine. Tuttavia, questa è sempre una pratica sconsigliabile, perché potrebbe risultare visibile nella riga di comando memorizzata nel sistema di gestione dei processi.

Figura 22.138. Midnight Commander incorpora le funzionalità di un cliente FTP.

Midnight Commander considera anche altri file system virtuali, come si vede nella documentazione originale: la pagina di manuale *mc(1)*.

22.17 Mcedit: programma integrato per la gestione dei file di testo

Midnight Commander dispone di un sistema integrato di gestione dei file di testo, denominato Mcedit. Uno dei problemi maggiori che incontra chi si avvicina a GNU/Linux, o a un altro sistema Unix, è la difficoltà nell'uso dei programmi tradizionali per la creazione e la modifica di file di testo. Questi richiedono normalmente la conoscenza di comandi espressi attraverso combinazioni di tasti, senza avere a disposizione un menù e senza poter usare i tasti funzionali. Mcedit risolve parzialmente questo problema, offrendo un funzionamento intuitivo per l'utente che proviene dall'esperienza Dos, quando si ha a disposizione un terminale adeguatamente raffinato, come una console virtuale di GNU/Linux.

Mcedit può funzionare anche se si utilizza un terminale poco dotato, ma in tal caso si ritorna a dover usare le solite combinazioni di tasti che annoiano alcuni utenti.

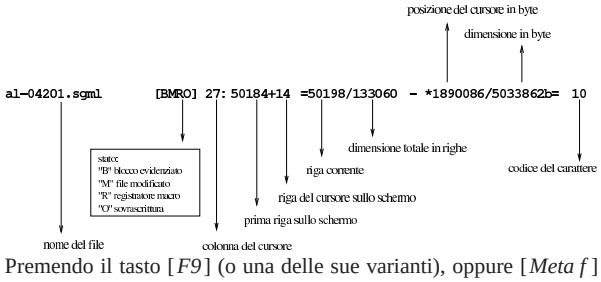
22.17.1 Avvio

L'avvio del sistema di gestione dei file di testo si ottiene semplicemente puntando il file desiderato con la barra di selezione di Midnight Commander e premendo il tasto [*F4*], oppure avviando il binario di Midnight Commander utilizzando il nome '**`mcedit`**' (si tratta normalmente di un collegamento simbolico a '**`mc`**').

Ci si inserisce in questo modo in un ambiente differente (anche se uniforme) per la modifica del file. In basso si notano alcuni riferimenti a tasti funzionali:

In alto si vede una riga di stato, dalla quale si conosce il nome del file aperto, lo stato del file, la colonna su cui si trova il cursore, il numero della prima riga che appare sullo schermo, il numero della riga relativa allo schermo, la riga assoluta in cui si trova il cursore, il numero totale di righe del file e altre informazioni, come si vede nella figura 22.140.

Figura 22.140. La riga di stato del sistema di gestione dei file di testo di Midnight Commander.



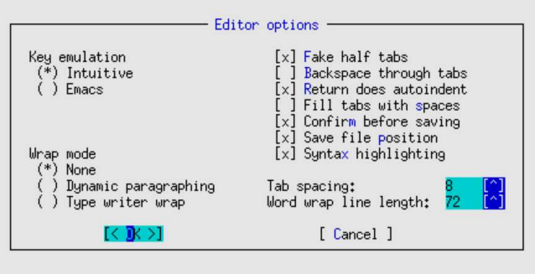
Premendo il tasto [*F9*] (o una delle sue varianti), oppure [*Meta f*] (in quanto la «f» è l'iniziale di «file»), la riga di stato viene sosti-

tuita da un menù a tendina che si può utilizzare in modo semplice e intuitivo.

22.17.2 Configurazione

La configurazione del sistema di gestione dei file di testo è accessibile attraverso il menù *Options*, in particolare merita attenzione la funzione *General*, come si vede in figura 22.141.

Figura 22.141. La maschera di configurazione generale.



Il significato delle varie opzioni dovrebbe essere evidente. In particolare, nel seguito viene mostrato l'uso della tastiera secondo l'emulazione «intuitiva».

Merita un po' di attenzione la gestione del carattere di tabulazione. Dal momento che lo standard generale prevede che la tabulazione dei file di testo sia ogni otto colonne, è opportuno che questa dimensione non venga alterata, mentre diviene conveniente l'opzione *fake half tab*. Questa permette di utilizzare il tasto [*Tab*] per inserire spaziature ridotte (di quattro caratteri) che poi vengono tradotte in pratica con tanti caratteri spazio (<*SP*>) quanti servono. Se però l'utente usa più volte la tabulazione, dove possibile viene utilizzato il carattere di tabulazione orizzontale (<*HT*>).

22.17.3 Comandi comuni

Mcedit è «intuitivo», in quanto si può usare senza dovere ricordare delle combinazioni di tasti. Quando serve qualcosa di più basta chiamare il menù e cercare tra le varie voci disponibili. Tuttavia, quando si utilizza attraverso un terminale non sufficientemente raffinato, si rischia di perdere l'uso di alcuni tasti, per cui si deve ripiegare sul solito sistema di combinazioni. La tabella 22.142 mostra l'elenco di alcuni comandi utili per lo spostamento del cursore e per la modifica del testo.

Tabella 22.142. Alcuni comandi per la navigazione e la modifica del testo.

Comando	Alternativa	Descrizione
[*Meta Esc*]	[*Esc*][*Esc*]	Termina l'attività sul file.
Caratteri normali		Inseriscono il testo corrispondente.
[*Tab*]	[*Ctrl i*]	A seconda della configurazione, inserisce una tabulazione.
Tasti freccia		Spostano il cursore nella direzione della freccia.
Tasti pagina		Spostano il cursore di una schermata in avanti o indietro.
[*Ctrl pagina-su*]		Sposta il cursore all'inizio del file.
[*Ctrl pagina-giù*]		Sposta il cursore alla fine del file.
[*Backspace*]	[*Ctrl h*]	Cancella il carattere a sinistra del cursore.
[*Canc*]	[*Ctrl d*]	Cancella il carattere in corrispondenza del cursore.
[*Invio*]	[*Ctrl j*]	Inserisce un'interruzione di riga.
[*Ctrl y*]		Elimina la riga su cui si trova il cursore.
[*Ctrl l*]		Ridisegna lo schermo.
[*Ins*]		Scambia tra sovrascrittura e inserimento del testo digitato.
[*Ctrl u*]		Annulla l'ultimo comando (compresa la singola digitazione).

22.17.4 Blocchi di testo e le funzionalità di taglia-copia-incolla

I blocchi di testo possono essere evidenziati, copiati, tagliati e incollati nello stesso modo utilizzato comunemente dai programmi fatti per il sistema Dos. Questo significa che tenendo premuto il tasto [*Maiuscole*] e utilizzando i tasti freccia, si ottiene la selezione del testo con trascinamento, fino a quando si rilascia nuovamente il tasto [*Maiuscole*]; la combinazione [*Ctrl Ins*] copia la selezione corrente, deselezionandola; la combinazione [*Maiuscole Ins*] incolla la selezione copiata precedentemente, nella posizione del cursore.

Tutto questo però, vale solo se si sta usando una console virtuale GNU/Linux; se si sta lavorando da un terminale differente, o una connessione remota, si usa il tasto [*F3*] (con le varianti eventuali dei tasti funzionali, già descritte per Midnight Commander) per segnalare l'inizio di una zona da marcare, quindi si sposta il cursore e infine si preme nuovamente [*F3*] per concludere la zona evidenziata.

In entrambi i casi, si potrebbe evitare la copia della zona selezionata, premendo semplicemente il tasto [*F5*] per incollare nella posizione del cursore, senza nemmeno perdere la selezione.

Per ritagliare l'area selezionata, si utilizza la combinazione [*Maiuscole Canc*]; poi si può incollare quanto ritagliato nel solito modo: [*Maiuscole Ins*].

Ciò che viene ritagliato, o copiato, viene inserito in un file temporaneo nella directory personale dell'utente: '`~/.mc/cedit/cooledit.clip`'. L'estensione del file suggerisce che si tratti di una *clipboard*. Questo meccanismo è molto importante, in quanto permette di incollare testo copiato o ritagliato con un'altra sessione di lavoro di Mcedit (attraverso Midnight Commander), purché appartenga allo stesso utente.

Mcedit consente anche l'utilizzo del mouse per delimitare un blocco di testo, anche quando si utilizza il programma all'interno di una finestra di terminale, nell'ambito del sistema grafico X.

Tabella 22.143. Taglia-copia-incolla.

Comando	Alternativa	Descrizione
[*Maiuscole freccia*] ... [*F3*]		Delimita un blocco di testo.
[*F5*]		Incolla una copia del testo evidenziato.
[*F6*]		Sposta il blocco evidenziato in corrispondenza del cursore.
[*F8*]		Cancella il blocco evidenziato.
[*Ctrl Ins*]		Copia la zona evidenziata nell'area temporanea, togliendo l'evidenziamento.
[*Maiuscole Canc*]		Ritaglia la zona evidenziata inserendola nell'area temporanea.
[*Maiuscole Ins*]		Incolla il contenuto dell'area temporanea nella posizione del cursore.

Infine, per eliminare un blocco di testo evidenziato, senza inserirlo nell'area temporanea, basta usare il tasto [*F8*].

22.17.5 Composizione del testo

Attraverso la configurazione di Mcedit, è possibile stabilire se si intende fare in modo che il testo sia mandato a capo automaticamente o meno. Per questo è anche necessario fissare la dimensione massima di colonne del testo.

In generale, dovrebbe essere conveniente evitare che Mcedit provveda da solo a dividere le righe mentre si digita il testo. Quando si vuole riallineare un paragrafo, cioè un blocco di righe di testo preceduto e seguito da una riga vuota, basta usare il comando [*Meta p*], oppure si può richiamare la voce corrispondente del menù *Edit*.

22.17.6 Macro

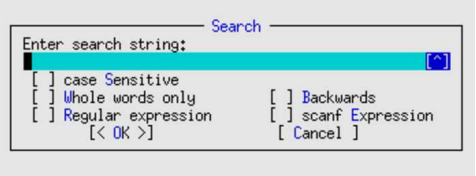
Attraverso la combinazione [*Ctrl r*], si inizia e si conclude la registrazione della pressione di una sequenza di tasti. Al termine viene richiesto di indicare un carattere; successivamente, per riprodurre quella sequenza, è sufficiente utilizzare la sequenza [*Meta a*][*x*],

dove l'ultimo tasto è appunto il carattere specificato in precedenza per memorizzare la macro.

22.17.7 Ricerche e sostituzioni

Dal menù *Search/Replace* si accede alle funzionalità di ricerca e di sostituzione del testo. La voce *Search*, a cui si accede facilmente anche con il tasto [*F7*], apre un riquadro come quello che si vede nella figura 22.144.

Figura 22.144. La maschera per la ricerca.



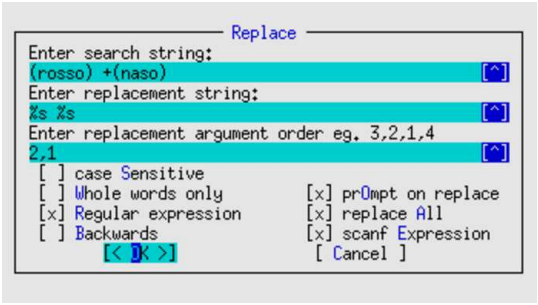
L'utilizzo delle caselle di selezione è abbastanza intuitivo. In particolare, è possibile utilizzare le espressioni regolari; inoltre, se funziona il mouse, è possibile selezionare il simbolo '**[^]**' con il quale si ottiene lo storico delle stringhe di ricerca già inserite.

La ricerca con sostituzione è un po' più complessa; in tal caso, la stringa di sostituzione può essere costante, o contenere anche un espressione da trattare come se fosse inserita in una funzione ***printf()***.

Purtroppo, tale funzionalità, pur essendo documentata e anche molto importante, potrebbe non essere operativa. Pertanto, se l'esempio che viene proposto non funziona, dipende dalla propria edizione di Mcedit.

Nella figura 22.145 si mostra un esempio in cui si cerca la stringa «rosso naso» e la si vuole sostituire con «naso rosso». Guardando la figura, si osservi il fatto che nell'espressione regolare di ricerca sono state delimitate le parole «naso» e «rosso» con delle parentesi tonde; a queste parole, ovvero alla stringa corrispondente secondo l'espressione regolare, si fa riferimento attraverso le metavariabili '**%s**', della funzione ***printf()***; tuttavia, si rende necessaria l'inversione nell'ordine di questi due parametri.

Figura 22.145. La maschera per la ricerca e la sostituzione, con un esempio.



In alternativa all'esempio mostrato, in cui si fa uso di un'espressione regolare per la ricerca (ammesso che la sostituzione funzioni come dovrebbe), si potrebbe usare un'espressione adatta alla funzione C ***scanf()***. In tal caso, l'opzione *Regular expression* andrebbe disattivata.

22.17.8 Collegamento «mcedit» e variabile di ambiente «EDITOR»

Si è accennato al fatto che si può ottenere immediatamente l'avvio di Mcedit avviando Midnight Commander con il nome '**mcedit**'. In tal caso sono valide la maggior parte delle opzioni che si potrebbero dare a '**mc**', con la differenza che si può aggiungere l'indicazione di un nome di file nella stessa riga di comando, così da poter creare o accedere immediatamente a un file di testo.


```
mcedit [opzioni] [file]
```

Se si apprezzano le caratteristiche di Mcedit, la presenza di questo collegamento permette di utilizzarlo anche al di fuori del funzionamento normale di Midnight Commander e di indicarlo come programma predefinito per la creazione e la modifica di file di testo.

22.17.9 Limiti di memoria

Di norma, Mcedit non è in grado di gestire file con una dimensione superiore a 64 Mibyte. Raramente può succedere di avere la necessità di gestire file di testo di dimensioni maggiori; in quei casi è necessario modificare uno o due valori nel file '`edit/edit.h`' dei sorgenti di Midnight Commander:

```
…
#define MAXBUFF 2048 /* il valore originale è 1024 */
…
…
#define START_STACK_SIZE 64 /* il valore originale è 32 */
…
```

Nel modo mostrato si raddoppiano i limiti, ma si osservi che il valore '**START_STACK_SIZE**' riguardante la pila per l'annullamento dei comandi, non deve essere ampliato necessariamente.

Eventualmente, per la ricompilazione di un pacchetto Debian, si veda la sezione 7.8.

22.18 Giochi e simili

Nei sistemi BSD e derivati si sono prodotti dei programmi, da usare attraverso uno schermo a caratteri, che sono da considerare come dei giochi o poco più. Questi programmi sono disponibili anche nei sistemi GNU e possono avere, soprattutto, un significato storico, ma vale comunque la pena di conoscerne alcuni.

Di per sé, un «gioco» è un programma come un altro, tuttavia, secondo la tradizione questi programmi vanno collocati nella gerarchia '`/usr/games/`', pertanto questa directory deve essere presente tra i percorsi di avvio degli eseguibili (la variabile di ambiente ***PATH***).

22.18.1 Banner

Il programma '**banner**'[51] serve a produrre un testo di grandi dimensioni, possibilmente da stampare attraverso una stampante a modulo continuo, per produrre un «manifesto», così come il nome del programma intende:

```
banner [opzioni] [-w n] [testo]
```

Il testo da ingrandire viene indicato alla fine della riga di comando; tuttavia, se manca, viene chiesto espressamente in modo interattivo. Come si vede dal modello, l'opzione più importante è '**-w**', con la quale si può specificare l'ampiezza massima da utilizzare per la stampa o la visualizzazione.

```
$ banner -w 40 ciao [Invio]
```

```
      #####
   ###########
  #############
 ##           ##
 #             #
 #             #
 #            ##
  #          ###
   ##
 #             #
 ################    ####
 ################    ####
 ################     ##
    ###
  #######   ##
 #########  ####
 ##    ##   # ##
 #     ##     #
 ##    #     ##
 ###############
  #############
 #

     #####
   ###########
  #############
 ##           ##
```

```
#          #
#          #
#          ##
####     ####
  ############
    ########
```

L'esempio mostra il funzionamento del programma per visualizzare la parola «ciao», usando un'ampiezza massima di 40 colonne di testo.

A seconda della versione, il programma '**banner**' potrebbe anche visualizzare il testo in orizzontale, rendendolo utile come un modo per evidenziare dei messaggi da generare attraverso degli script.

22.18.2 Arithmetic

Il programma '**arithmetic**'[52] permette di eseguire degli esercizi di aritmetica elementare:

```
arihmetic [-o {+|-|x|/}…] [-r n]
```

Se si utilizza '**arithmetic**' senza argomenti, si ottengono degli esercizi molto semplici su somme e sottrazioni; altrimenti, con l'opzione '**-o**' è possibile specificare il tipo di operazioni su cui puntare l'attenzione (somma, sottrazione, moltiplicazione e divisione). L'opzione '**-r**' consente di indicare il numero intero massimo da usare nelle operazioni (a seconda dei casi può trattarsi degli operandi o dei risultati che si ottengono, secondo un ragionevole buon senso).

```
$ arithmetic -o +-x/ [Invio]
```

In questo caso, si vogliono eseguire esercizi su tutte le quattro operazioni, nell'intervallo numerico predefinito:

```
61 / 7 = 8 [Invio]

  Right!

0 x 1 = 0 [Invio]

  Right!

94 / 9 = 10 [Invio]

  Right!

9 + 6 = 16 [Invio]

  What?

15 [Invio]

  Right!

9 x 7 = 63 [Invio]

  Right!

14 - 6 = [Ctrl c]

$ arithmetic -o +-x/ -r 2 [Invio]
```

In questo caso, con l'opzione '**-r 2**' si vuole che il programma si concentri sui valori da zero a due.

22.18.3 Primes

Il programma '**primes**'[53] elenca i numeri primi a partire da un certo valore o appartenenti a un certo intervallo:

```
primes [inizio [fine]]
```

Se si utilizza '**primes**' senza argomenti, questo attende dallo standard input il valore iniziale e quindi inizia a calcolare i numeri primi da quel valore fino a un certo numero molto grande; altrimenti, i calcoli si limitano all'intervallo indicato negli argomenti.

```
$ primes 100 130 [Invio]
```

L'esempio serve a ottenere i numeri primi compresi tra 100 e 130. Ecco cosa si ottiene:


```
101
103
107
109
113
127
```

22.18.4 POM

Il programma '**pom**',[54] ovvero *Phase of moon*, genera la descrizione della fase lunare del momento presente o di un altro momento specificato nella riga di comando:

```
pom [[[[[ss]aa]mm]gg]hh]
```

```
$ pom [Invio]
```

In questo caso si ottiene la fase attuale:

```
The Moon is Waning Crescent (8% of Full)
```

Se invece si specifica una data:

```
$ pom 2010123100 [Invio]

Thu 2010 Dec 30 23:00:00 (CET): ↩
↪The Moon will be Waning Crescent (21% of Full)
```

22.18.5 Worms

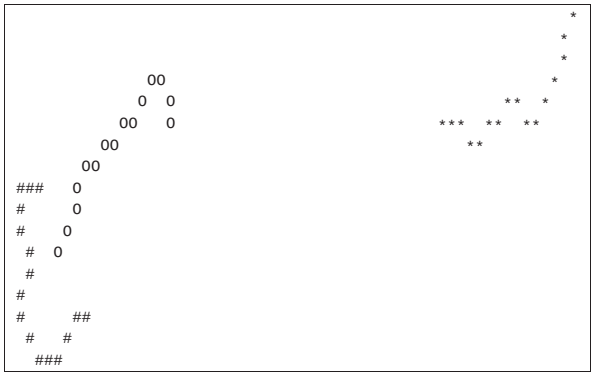
Il programma '**worms**',[55] è una specie di salva-schermo per terminali a carattere, che disegna il percorso di alcuni «vermi». Nella riga di comando sono previste delle opzioni, dove è praticamente obbligatorio l'uso di '**-d *n***', per specificare un ritardo:

```
worms [opzioni]
```

Ecco come può essere usato in una console di un sistema GNU/Linux:

```
$ worms -d 100 [Invio]
```

Ecco come potrebbe presentarsi lo schermo dopo un po':



Per aumentare la quantità di vermi sullo schermo, basta usare l'opzione '**-n *n***':

```
$ worms -d 100 -n 5 [Invio]
```

In tal caso i vermi sono cinque.

22.18.6 Rain

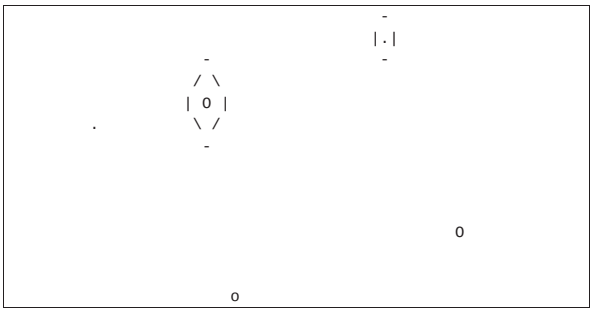
Il programma '**rain**',[56] è una specie di salva-schermo per terminali a carattere, che simula sullo schermo il gocciolio della pioggia:

```
rain [-d n]
```

L'opzione '**-d**' serve a richiedere un ritardo nel ciclo di iterazione, che diventa praticamente obbligatorio per poter distinguere qualcosa:

```
$ rain -d 100 [Invio]
```

Ecco come potrebbe presentarsi lo schermo durante il funzionamento:



22.18.7 Fortune

Il programma '**fortune**'[57] serve a mostrare degli «epigrammi» (delle «massime», ma spesso sono semplici barzellette), raccolti in file appositi, che di norma accompagnano il programma stesso:

```
fortune [opzioni]
```

Il programma viene usato normalmente senza opzioni, per ottenere una massima a caso:

```
$ fortune [Invio]

  Extreme feminine beauty is always disturbing.
                  -- Spock, "The Cloud Minders", stardate 5818.4
```

22.18.8 Worm

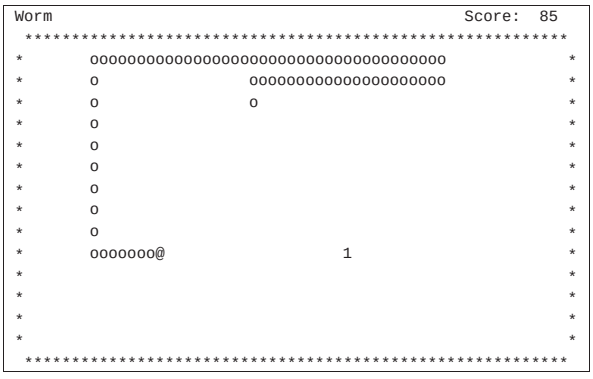
Il programma '**worm**'[58] è il gioco del verme che cresce allungandosi. Lo scopo del gioco è quello di «mangiare» dei numeri, raggiungendoli con la testa del verme, senza che il verme tocchi se stesso o i bordi dello schermo:

```
worm [dimensione]
```

Il verme è costituito da una sequenza di caratteri «o» e la testa la si distingue perché è rappresentata da «@». Inizialmente il verme si presenta lungo otto caratteri in tutto, a meno di utilizzare l'opzione numerica, che può servire per riprendere il gioco con un verme più grande.

```
$ worm [Invio]
```

Ecco come si presenta il gioco dopo un po' di tempo che viene utilizzato, quando ha raggiunto la dimensione di 85 caratteri:



Per dirigere il verme dovrebbe essere possibile usare i tasti freccia, ma rimane la possibilità di usare i tasti [*h*], [*j*], [*k*], e [*l*], come con il programma VI.

22.18.9 Tetris

Il programma '**tetris-bsd**'[59] è il gioco del «tetris», nella versione per terminali a caratteri.

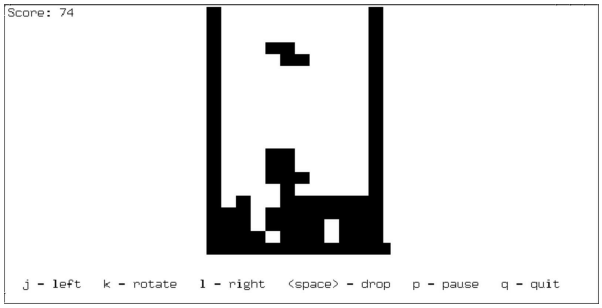
```
tetris [opzioni]
```

```
tetris-bsd [opzioni]
```

Attraverso le opzioni è possibile incrementare il livello di gioco e attivare l'anteprima del pezzo successivo. Si veda eventualmente la pagina di manuale *tetris(6)* o *tetris-bsd(6)*.

```
$ tetris-bsd [Invio]
```

Ecco come si presenta il gioco dopo un po' di tempo che viene utilizzato:



Per dirigere i pezzi si usano i tasti [*j*], [*k*] e [*l*], che servono rispettivamente per spostare a sinistra, ruotare e spostare a destra. Per fare una pausa si usa il tasto [*p*], mentre per far scendere immediatamente un pezzo si usa la [*barra-spaziatrice*].

22.19 Alternative nella distribuzione Debian

La distribuzione GNU/Linux Debian organizza una serie di collegamenti simbolici riferiti a programmi di uso comune; per esempio, in questo modo esiste il programma eseguibile virtuale '**x-www-browser**', che fa riferimento al navigatore (per esempio '**mozilla**'), oppure esiste il programma virtuale '**vi**' che fa riferimento a '**vim**', '**nvi**' o simili.

Questo sistema ha il vantaggio di offrire un nome standard ai programmi, così da avere la certezza di trovare sempre ciò che serve attraverso un nome che non cambi nel tempo. Ovviamente occorre avere l'accortezza di aggiornare questi collegamenti simbolici in base a ciò che si installa e si disinstalla.

Anche se si tratta di una struttura specifica della distribuzione Debian, ciò non toglie che possa essere usato un metodo del genere anche in altre distribuzioni, oppure che si possa aggiungere da soli qualcosa di simile.

22.19.1 Organizzazione particolare della distribuzione Debian

La distribuzione Debian è organizzata in modo tale da inserire i collegamenti simbolici da aggiornare nella directory '`/etc/alternatives/`'. L'elenco di alcuni collegamenti comuni appare nella tabella 22.166.

La directory '`/etc/alternatives/`' non appare nell'elenco dei percorsi di avvio degli eseguibili (contenuto nella variabile di ambiente ***PATH***), pertanto, nelle directory '`/bin/`', '`/usr/bin/`' ed eventualmente in altre simili, si trovano altri collegamenti simbolici che puntano a quelli previsti nella directory '`/etc/alternatives/`'. Per esempio, '`/usr/bin/vi`' punta a '`/etc/alternatives/vi`', il quale, a sua volta, potrebbe puntare a '`/usr/bin/nvi`' o altro programma compatibile con VI.

Oltre al problema degli eseguibili, la distribuzione Debian fa in modo che anche i file delle pagine di manuale siano inseriti in questo

meccanismo. Pertanto, per continuare l'esempio già proposto a proposito di VI, '`/usr/share/man/man1/vi.1.gz`' punta a '`/etc/alternatives/vi.1.gz`', il quale potrebbe puntare a sua volta a '`/usr/share/man/man1/nvi.1.gz`'.

La gestione dei collegamenti simbolici contenuti nella directory '`/etc/alternatives/`' può essere manuale, oppure automatica; in generale può essere controllata attraverso il programma '**`update-alternatives`**'. Il modello sintattico seguente mostra solo alcuni casi di utilizzo comuni:

```
update-alternatives [opzioni_comuni] comando nome
```

Tabella 22.162. Alcuni casi comuni.

Sintassi	Descrizione
`update-alternatives` [*opzioni*] `--display` *nome*	Mostra lo stato del collegamento simbolico indicato come argomento finale.
`update-alternatives` [*opzioni*] `--list` *nome*	Mostra l'elenco delle alternative possibili riferite al collegamento simbolico indicato come argomento finale.
`update-alternatives` [*opzioni*] `--config` *nome*	Consente di decidere interattivamente come dirigere il collegamento simbolico indicato.
`update-alternatives` [*opzioni*] `--auto` *nome*	Riporta allo stato automatico la gestione del collegamento simbolico indicato come argomento finale.

Dalla descrizione della tabella si vede che un riferimento a un programma può essere gestito in modo automatico, oppure manuale. Se si interviene a mano nei collegamenti simbolici della directory '`/etc/alternatives/`', questi vengono riconosciuti come riferimenti gestiti manualmente. Ecco cosa si potrebbe vedere a proposito di uno di questi riferimenti:

```
# update-alternatives --display x-www-browser [Invio]

x-www-browser - status is auto.
 link currently points to /usr/bin/galeon
/usr/bin/mozilla - priority 80
 slave x-www-browser.1.gz: /usr/share/man/man1/mozilla.1.gz
/usr/bin/galeon - priority 120
 slave x-www-browser.1.gz: /usr/share/man/man1/galeon.1.gz
/usr/bin/epiphany - priority 85
 slave x-www-browser.1.gz: /usr/share/man/man1/epiphany.1.gz
/usr/bin/konqueror - priority 100
Current 'best' version is /usr/bin/galeon.
```

Volendo impostare manualmente il navigatore predefinito in modo che corrisponda a Mozilla, ecco come bisognerebbe procedere:

```
# update-alternatives --configure x-www-browser [Invio]

There are 4 alternatives which provide 'x-www-browser'.

  Selection    Alternative
-----------------------------------------------
      1        /usr/bin/mozilla
*+    2        /usr/bin/galeon
      3        /usr/bin/epiphany
      4        /usr/bin/konqueror

Press enter to keep the default[*], or type selection
number: 1 [Invio]

Using '/usr/bin/mozilla' to provide 'x-www-browser'.
```

Per riportare tutto allo stato automatico:

```
# update-alternatives --auto x-www-browser [Invio]
```


Tabella 22.166. Alcune alternative Debian che fanno riferimento a nomi di programmi non comuni.

Comando	Descrizione
`editor`	Un programma per la creazione e la modifica dei file di testo.
`view`	Un programma per la visualizzazione di file comuni.
`pager`	Un programma per scorrere il contenuto di un file sullo schermo (come '**`more`**' o '**`less`**').
`www-browser`	Navigatore per i terminali non grafici.
`x-session-manager`	Un gestore di sessione per il sistema grafico X.
`x-terminal-emulator`	Un terminale grafico per X.
`x-window-manager`	Un gestore di finestre per il sistema grafico X.
`x-www-browser`	Navigatore grafico.

22.19.2 Organizzare manualmente un sistema di alternative

Se piace l'idea delle alternative, si può realizzare un sistema simile anche in modo manuale, senza tante raffinatezze: è sufficiente predisporre una directory con i collegamenti che servono, mettendo questa directory al primo posto dei percorsi di ricerca. Supponendo che si tratti della directory '`/etc/bin/`':

```
PATH="/etc/bin:$PATH"
export $PATH
```

Le istruzioni appena mostrate potrebbero risiedere nel file '`/etc/profile`', supponendo che sia utilizzata una shell POSIX o una shell Bourne.

22.20 Riferimenti

- *IEEE P1003.2 Draft 11.2*, September 1991, *bc - Arbitrary-precision arithmetic language*, *http://www.nic.funet.fi/pub/doc/posix/p1003.2/d11.2/4.3*
- *vi (and clone) docs*, *http://www.rru.com/~meo/useful/vi/*

[1] **GNU core utilities** GNU GPL

[2] **GNU core utilities** GNU GPL

[3] **GNU core utilities** GNU GPL

[4] **GNU core utilities** GNU GPL

[5] **util-linux: rev** UCB BSD

[6] **Dog** GNU GPL

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

[8] **GNU core utilities** GNU GPL

[9] **GNU core utilities** GNU GPL

[10] **column** UCB BSD

[11] **colrm** UCB BSD

[12] Dei file di testo contenenti codici di spostamento di mezza riga, in avanti o indietro, possono essere generati da programmi come Nroff e Tbl (capitolo 25).

[13] **col** UCB BSD

[14] **colcrt** UCB BSD

[15] **ul** UCB BSD

[16] **GNU core utilities** GNU GPL

[17] **GNU core utilities** GNU GPL

[18] **GNU core utilities** GNU GPL

[19] **GNU core utilities** GNU GPL

[20] **GNU core utilities** GNU GPL

[21] **GNU core utilities** GNU GPL

[22] **GNU core utilities** GNU GPL

[23] **GNU core utilities** GNU GPL

[24] **GNU core utilities** GNU GPL

[25] **GNU core utilities** GNU GPL

[26] **GNU core utilities** GNU GPL

[27] **GNU core utilities** GNU GPL

[28] **GNU core utilities** GNU GPL

[29] **GNU core utilities** GNU GPL

[30] Purtroppo non si può ottenere il contrario con '`tr`'.

[31] **GNU core utilities** GNU GPL

[32] **GNU core utilities** GNU GPL

[33] **GNU core utilities** GNU GPL

[34] **GNU core utilities** GNU GPL

[35] **BSD utils** UCB BSD

[36] **hexcat** software libero con licenza speciale

[37] **GNU diffutils** GNU GPL

[38] **GNU Patch** GNU GPL

[39] **GNU core utilities** GNU GPL

[40] **GNU core utilities** GNU GPL

[41] uno, zero, zero in base venti, corrisponde a 400.

[42] **GNU BC** GNU GPL

[43] **Elvis < 2.0** software libero

[44] **Elvis ≥ 2.1** software non libero: licenza Artistic

[45] **Vim** software libero con licenza speciale

[46] **nVI** software libero

[47] Le sigle '`~E`', '`~J`', ecc. rappresentano le lettere accentate della codifica utilizzata nei sistemi Dos. Per scrivere un file del genere, occorrono due fasi: una in un ambiente che accetti la codifica Latin 1 e l'altra in Dos.

[48] **GNU Midnight Commander** GNU GPL

[49] Data la funzione particolare che ha il tasto [*Esc*], come sostituto del tasto [*Meta*], si può comprendere il motivo per il quale, spesso, per annullare un'operazione occorre premere due volte il tasto [*Esc*].

[50] Le finestre di dialogo contengono diversi tipi di elementi e la conferma o l'annullamento si ottengono selezionando una voce che riproduce una sorta di tasto virtuale.

[51] **BSD utils** UCB BSD

[52] **BSD games** UCB BSD

[53] **BSD games** UCB BSD

[54] **BSD games** UCB BSD

[55] **BSD games** UCB BSD

[56] **BSD games** UCB BSD

[57] **Fortune** UCB BSD

[58] **BSD games** UCB BSD

[59] **BSD games** UCB BSD