Appendice C. Una breve introduzione a Sed e Awk

Sommario

Questa è una brevissima introduzione alle utility di elaborazione di testo **sed** e **awk**. Qui vengono trattati solamente alcuni comandi di base, ma che saranno sufficienti per comprendere i semplici costrutti sed e awk presenti negli script di shell.

sed: editor non interattivo di file di testo

awk: linguaggio per l'elaborazione di modelli orientato ai campi, con una sintassi simile a quella del C

Nonostante le loro differenze, le due utility condividono una sintassi d'invocazione simile, entrambe fanno uso delle Espressioni Regolari, entrambe leggono l'input, in modo predefinito, dallo `stdin` ed entrambe inviano i risultati allo `stdout`. Sono strumenti UNIX ben collaudati e funzionano bene insieme. L'output dell'una può essere collegato, per mezzo di una pipe, all'altra e le loro capacità combinate danno agli script di shell parte della potenza di Perl.

Un'importante differenza tra le due utility è che mentre gli script di shell possono passare facilmente degli argomenti a sed, è più complicato fare la stessa cosa con awk (vedi Esempio 33-5 e Esempio 9-24).

C.1. Sed

Sed è un editor di riga non interattivo. Riceve un testo come input, o dallo `stdin` o da un file, esegue alcune operazioni sulle righe specificate, una alla volta, quindi invia il risultato allo `stdout` o in un file. Negli script di shell, sed è, di solito, una delle molte componenti di una pipe.

Sed determina le righe dell'input, su cui deve operare, tramite un *indirizzo* che gli è stato passato. [1] Questo indirizzo può essere rappresentato sia da un numero di riga sia da una verifica d'occorrenza. Ad esempio, *`3d`* indica a sed di cancellare la terza riga dell'input, mentre *`/windows/d`* segnala a sed che si vogliono cancellare tutte le righe dell'input contenenti l'occorrenza "windows".

Di tutte le operazioni a disposizione di sed, vengono focalizzate, in primo luogo, le tre più comunemente usate. Esse sono **p**rint (visualizza allo `stdout`), **d**elete (cancella) e **s**ubstitute (sostituisce).

Tabella C-1. Operatori sed di base

Operatore	Nome	Effetto
`[indirizzo]/p`	print	Visualizza [l'indirizzo specificato]
`[indirizzo]/d`	delete	Cancella [l'indirizzo specificato]
`s/modello1/modello2/`	substitute	Sostituisce in ogni riga la prima occorrenza della stringa modello1 con la stringa modello2
`[indirizzo]/s/modello1/modello2/`	substitute	Sostituisce, in tutte le righe specificate in *`indirizzo`*, la prima occorrenza della stringa modello1 con la stringa modello2
`[indirizzo]/y/modello1/modello2/`	transform	sostituisce tutti i caratteri della stringa modello1 con i corrispondenti caratteri della stringa modello2, in tutte le righe specificate da *`indirizzo`* (equivalente di **tr**)
`g`	global	Agisce su *tutte* le verifiche d'occorrenza di ogni riga di input controllata

Se l'operatore `g` (*global*) non è accodato al comando *substitute*, la sostituzione agisce solo sulla prima verifica d'occorrenza di ogni riga.

Sia da riga di comando che in uno script di shell, un'operazione sed può richiedere il quoting e alcune opzioni.

```
sed -e '/^$/d' $nomefile
#  L'opzione -e indica che la stringa successiva deve essere interpretata come
#+ un'istruzione di editing.
#  (se a "sed" viene passata un'unica istruzione, "-e" è facoltativo.)
#  Il quoting "forte" ('') protegge i caratteri speciali delle ER, presenti
#+ nell'istruzione, dalla reinterpretazione da parte dello script.
# (Questo riserva solo a sed l'espansione delle ER.)
#
# Agisce sul testo del file $nomefile.
```

In certi casi, un comando di editing **sed** non funziona in presenza degli apici singoli.

```
nomefile=file1.txt
modello=INIZIO

  sed "/^$modello/d" "$nomefile"  # Funziona come indicato.
# sed '/^$modello/d' "$nomefile"    dà risultati imprevisti.
#        In questo esempio, il quoting forte (' ... '),
#+       impedisce a "$modello" di espandersi a "INIZIO".
```

Sed utilizza l'opzione `-e` per indicare che la stringa che segue è un'istruzione, o una serie di istruzioni. Se la stringa contiene una singola istruzione, allora questa opzione può essere omessa.

```
sed -n '/xzy/p' $nomefile
#  L'opzione -n indica a sed di visualizzare solo quelle righe che verificano
#+ il modello.
#  Altrimenti verrebbero visualizzate tutte le righe dell'input.
#  L'opzione -e, in questo caso, non sarebbe necessaria perché vi è una sola
```

```
#+ istruzione di editing.
```

Tabella C-2. Esempi di operatori sed

Notazione	Effetto
`8d`	Cancella l'ottava riga dell'input.
`/^$/d`	Cancella tutte le righe vuote.
`1,/^$/d`	Cancella dall'inizio dell'input fino alla prima riga vuota compresa.
`/Jones/p`	Visualizza solo le righe in cui è presente "Jones" (con l'opzione -n).
`s/Windows/Linux/`	Sostituisce con "Linux" la prima occorrenza di "Windows" trovata in ogni riga dell'input.
`s/BSOD/stabilità/g`	Sostituisce con "stabilità" tutte le occorrenze di "BSOD" trovate in ogni riga dell'input.
`s/ *$//`	Cancella tutti gli spazi che si trovano alla fine di ogni riga.
`s/00*/0/g`	Riduce ogni sequenza consecutiva di zeri ad un unico zero.
`/GUI/d`	Cancella tutte le righe in cui è presente "GUI".
`s/GUI//g`	Cancella tutte le occorrenze di "GUI", lasciando inalterata la parte restante di ciascuna riga.

Sostituire una stringa con un'altra di lunghezza zero (nulla) equivale a cancellare quella stringa nella riga di input. Questo lascia intatta la parte restante della riga. L'espressione `s/GUI//` applicata alla riga

```
Le parti più importanti di ogni applicazione sono le sue GUI e gli effetti sonori
```

dà come risultato

```
Le parti più importanti di ogni applicazione sono le sue e gli effetti sonori
```

La barra inversa costringe il comando di sostituzione **sed** a continuare sulla riga successiva. L'effetto è quello di usare il

carattere di *a capo* alla fine della prima riga come *stringa di sostituzione*.

```
s/^  */\
/g
```

In questo modo, tutti gli spazi che si trovano all'inizio della riga vengono sostituiti con un carattere di a capo. Il risultato finale è la sostituzione di tutte le indentazioni dei paragrafi con righe vuote poste tra gli stessi paragrafi.

Un indirizzo seguito da una, o più, operazioni può richiedere l'impiego della parentesi graffa aperta e chiusa, con un uso appropriato dei caratteri di a capo.

```
/[0-9A-Za-z]/,/^$/{
/^$/d
}
```

Questo cancella solo la prima di ogni serie di righe vuote. Potrebbe essere utile per effettuare la spaziatura singola di un file di testo mantenendo, però, la/e riga/he vuota/e tra i paragrafi.

L'usuale delimitatore usato da *sed* è la /. Tuttavia, *sed* consente anche altri delimitatori, come %. Questo torna utile quando la / è contenuta nella stringa di sostituzione, come nei percorsi dei file. Vedi Esempio 10-9 e Esempio 15-29.

La via più rapida per effettuare una spaziatura doppia di un file di testo è `sed G nomefile`.

Per esempi che illustrano l'uso di sed negli script di shell, vedi:

1. Esempio 33-1

2. Esempio 33-2
3. Esempio 15-3
4. Esempio A-2
5. Esempio 15-15
6. Esempio 15-24
7. Esempio A-12
8. Esempio A-17
9. Esempio 15-29
10. Esempio 10-9
11. Esempio 15-43
12. Esempio A-1
13. Esempio 15-13
14. Esempio 15-11
15. Esempio A-10
16. Esempio 18-12
17. Esempio 15-16
18. Esempio A-29

Per una spiegazione più ampia di sed, si controllino i relativi riferimenti in *Bibliografia*.

Note

[1] Se non viene specificato alcun indirizzo, sed, in modo predefinito, considera *tutte* le righe.
