

Sed & Awk

Stream EDitor (sed)

- sed: editor di linea che non richiede l'interazione con l'utente
- sed può filtrare l'input che riceve da un file o una pipe
- La sintassi di sed NON definisce un output:
 - L'output viene inviato allo standard output e può essere rediretto
- sed NON modifica l'input

Stream EDitor (sed)

SYNOPSIS

sed [-an] command [file ...]

[-an] [-e command] [-f command_file] [file ...]

- sed legge i file specificati, oppure lo standard input se non specificati file;
- modifica l'input come specificato da una lista di comandi;
- L'input è quindi scritto sullo standard output.

-n di default ogni linea e' replicata sullo standard output dopo l'applicazione dei comandi. Opzione -n elimina questo.

Comandi sed

Alcuni comandi:

- a\ “Append” di testo al di sotto della riga corrente
- c\ Modifica il testo della riga corrente
- d Cancella testo
- i\ Inserisci testo al di sopra della riga corrente
- p Stampa testo
- r Legge un file
- s Cerca e modifica testo

Comandi Sed

La forma del comando sed e' la seguente:

[address[,address]]function[arguments]

Normalmente sed:

1. copia ciclicamente una linea di input in un pattern space,
2. applica tutti i comandi con address selezionati dal pattern space,
3. copia il pattern space nell standard output, aggiungendo newline,
4. quindi cancella il pattern space.

Indirizzo: [address[,address]]

L'**indirizzo** non e' richiesto, ma se specificato deve essere:

1. un numero (che conta linee di input nei file di input),
 2. un carattere ``\$'' per l'ultima line di input
 3. oppure un address di contesto (espressione regolare preceduta o seguita da un delimitatore).
- Una linea di comando senza **indirizzo** seleziona ogni pattern space.
 - Una linea di comando con un **indirizzo** seleziona ogni pattern space dato dall'address.
 - Una linea di comando con due **indirizzo** seleziona il range inclusivo del primo pattern space tra i due primo indirizzi.

Sed

- Se non si specificano azioni, sed stampa sullo standard output le linee in input, lasciandole inalterate
- Se non viene specificato un **indirizzo** o un intervallo di indirizzi di linea su cui eseguire l'azione, quest'ultima viene applicata a tutte le linee in input.
- Gli indirizzi di linea si possono specificare come **numeri** o **espressioni regolari**.
- Se vi è più di un'azione (**comandi multipli**), esse possono essere specificate sulla riga di comando precedendo ognuna con l'opzione -e, oppure possono essere lette da un file esterno specificato sulla linea di comando con l'opzione -f.

Esempio

Iso:~>cat esempio

1 Questo e' un esempio.

2 Questa riga contiene un errore

3 Un altro errore in questa riga

4 Questa riga e' corretta

5 Questa riga contiene un altro errore.

Iso:~>grep errore esempio

2 Questa riga contiene un errore

3 Un altro errore in questa riga

5 Questa riga contiene un altro errore.

Comando stampa

Iso:~>sed '/erore/p' esempio

- 1 Questo e' un esempio.
- 2 Questa riga contiene un errore
- 2 Questa riga contiene un errore
- 3 Un altro errore in questa riga
- 3 Un altro errore in questa riga
- 4 Questa riga e' corretta
- 5 Questa riga contiene un altro errore.
- 5 Questa riga contiene un altro errore.

Iso:~>sed -n '/erore/p' esempio

- 2 Questa riga contiene un errore
- 3 Un altro errore in questa riga
- 5 Questa riga contiene un altro errore.

sed '/espressione/p' file

Stampa tutte le linee, quelle che contengono la stringa si ripetono

Per stampare solo le linee che contengono la stringa si usa l'opzione -n

Comando Cancella

Iso:~>sed '/errore/d' esempio sed '/espressione/d' file

1 Questo e' un esempio.

4 Questa riga e' corretta

Il comando **d** porta ad escludere linee dalla visualizzazione

Iso:~>sed -n '/^Questo.*esempio.\$/d' example

2 Questa riga contiene un errore

3 Un altro errore in questa riga

4 Questa riga e' corretta

5 Questa riga contiene un altro errore.

Escluse le linee che iniziano con una stringa e terminano con un'altra

Selezione di un range di righe

```
sed '[add1,[add2]]com'
```

Iso:~>sed '2,4d' esempio

(Cancella righe tra
2,4)

1 Questo e' un esempio.

5 Questa riga contiene un altro errore.

Iso:~>sed '3,\$d' esempio

(Cancella righe
tra 3 e ultima \$)

1 Questo e' un esempio.

2 Questa riga contiene un errore

Ricerca e Sostituzione

Iso:~>sed 's/erore/errore/g' esempio

- 1 Questo e' un esempio.
- 2 Questa riga contiene un **erore**
- 3 Un altro **erore** in questa riga
- 4 Questa riga e' corretta
- 5 Questa riga contiene un altro **erore**.

Iso:~>sed 's/^/> /g' esempio

- > 1 Questo e' un esempio.
- > 2 Questa riga contiene un **erore**
- > 3 Un altro **erore** in questa riga
- > 4 Questa riga e' corretta
- > 5 Questa riga contiene un altro **erore**

Sostituzione

```
's/{old value}/{new value}/'
```

Sostituzione globale

```
's/{old value}/{new value}/g'
```

Ricerca e Sostituzione

```
Iso:~>sed -e 's/erore/errore/g'  
-e 's/^/> /g' esempio
```

Multiple Changes

- > 1 Questo e' un esempio.
- > 2 Questa riga contiene un **errore**
- > 3 Un altro **errore** in questa riga
- > 4 Questa riga e' corretta
- > 5 Questa riga contiene un altro **errore**.

Elementi di Awk

- Funzione awk cerca su file linee o altre unità di testo che contengono pattern;
- Quando una linea corrisponde ad un pattern, azioni speciali vengono eseguiti sulla linea.
- In awk i programmi sono "data-driven": descrivi cosa cerchi, poi esegui;
- Il programma e' definito da un insieme di regole;
- Ogni regola e': azione da fare trovato il pattern.

Elementi di awk

- awk suddivide ogni linea in una sequenza di campi delimitati da separatori
 - *I separatori di default sono uno (o piu') spazi o caratteri di tabulazione.*
- Le variabili $\$1$, $\$2$, ..., $\$n$ identificano i primi n campi riconosciuti da awk
- La variabile $\$0$ contiene la riga in esame
- Sintassi:
 - ***awk '<programma>' <input-file1> <input-file2>...***
 - ***awk -f <file-script> <input-file> <input-file2>...***

Esempio

Iso:~>df display free disk space

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/hda5	16682168	9932136	5902608	63%	/
none	256840	0	256840	0%	/dev/shm

Iso:~>df | awk '{ print \$1,\$2, \$3}'

Filesystem	1K-blocks	Used
/dev/hda5	16682168	9932556
none	256840	0

Le variabili \$1, \$2, \$3, ..., \$N contengono il primo, secondo, terzo ... ultimo campo di una linea di input. La variabile \$0 (zero) contiene la linea intera.

Selezione di non-matching lines

Iso:~>df | grep -v Filesystem | awk '{ print "La partizione :"\$1 "\t e usata al "\$5}'

La partizione :/dev/hda5 e usata al 63%

La partizione :none e usata al 0%

awk ed Espressioni Regolari

- E' possibile combinare espressioni regolari e programmi awk utilizzando la seguente sintassi:
 - awk '<espressione>{<programma>}' <file>

- Esempio:

```
Iso:~>df | awk '/dev/hd/ { print "La partizione :"$1 "\t e usata al "$5}'
```

La partizione :/dev/hda5 e usata al 63%

- Il programma viene eseguito solo sulle righe che corrispondono al pattern dell'espressione regolare

awk: BEGIN ed END

Gli statement BEGIN ed END consentono eseguire operazioni prima e dopo il corpo del comando

```
Iso:~>df | awk 'BEGIN {print "Elenco partizioni"} /dev/hd/ { print  
"La partizione :"$1 "\t e usata al "$5} END {print "Fine  
Report\n"}'
```

Elenco partizioni

La partizione :/dev/hda5 e usata al 63%

Fine Report

awk scripts

- E' possibile definire script awk
 - L'esempio precedente

```
Iso:~>df | awk 'BEGIN {print "Elenco partizioni"} /dev/hd/ { print "La partizione :"$1 "\t e usata al "$5} END {print "Fine Report\n"}'
```

- Diventa:

```
BEGIN {print "Elenco partizioni"}
```

```
/dev/hd/ { print "La partizione :"$1 "\t e usata al "$5}
```

```
END {print "Fine Report\n"}
```

- Se il nome dello script e' report.awk

```
Iso:~>df |awk -f report.awk
```

```
Elenco partizioni
```

```
La partizione :/dev/hda5      e usata al 63%
```

```
Fine Report
```

awk: le variabili

Awk usa molte variabili, alcune editabili, altre read-only.

- **La variabile FS** (Field Separator) identifica il separatore di input
 - Default spazi o tab
- **La variabile OFS** (Output Field Separator) identifica il separatore di output
 - Default spazio
- **La variabile ORS** (Output Record Separator) identifica il separatore di “record”
 - Default \n
- E' possibile modificare il valore di queste variabili
 - **BEGIN { FS=";" ; OFS="---"; ORS=">\n<-" }**

awk: le variabili

- La variabile **NR** contiene il numero di record processati
 - Viene incrementata automaticamente
- Ogni riferimento ad una variabile non definita comporta la creazione della stessa e la sua inizializzazione a ""
 - I riferimenti successivi utilizzeranno il valore corrente della stessa

Esempio

lso:~>cat somma.awk

```
BEGIN {  
  FS=":";  
  print "Calcolo Subtotali e Totale"}  
{  
  subtotale=$1*$2;  
  print "Subtotale per " $3 "="subtotale;  
  totale = totale+subtotale;  
}  
END {  
  print "Totale ="totale  
}
```

lso:~>cat dati.txt

```
100:2:Cliente 1  
200:8:Cliente 2  
500:2:Cliente 3
```

Esempio

Iso:~>cat dati.txt

100:2:Cliente 1

200:8:Cliente 2

500:2:Cliente 3

Iso:~>awk -f somma.awk dati.txt

Calcolo Subtotali e Totale introiti

Subtotale per Cliente 1=200

Subtotale per Cliente 2=1600

Subtotale per Cliente 3=1000

Totale =2800

Output formattato

- awk consente di formattare l'output utilizzando la funzione printf (invece della funzione print)
- Sintassi: **printf formato, item1, item2,...**

- Esempio:

```
iso:~>awk 'BEGIN {w=5; p=3; s="abc"; printf "%d %4.3f %s\n",w,p,s}'
```

```
5 3.000 abc
```

- Il “formato” include **%d, %f, %c, %o, %x...**
- Nota: Lo statement BEGIN consente di eseguire programmi awk SENZA specificare un input (file o redirectione)

Redirezione in awk

- E' possibile utilizzare gli operatori di redirezione in script awk.
- **print items > nomefile**
 - awk '{ print \$2 > "phone-list"; print \$1 > "name-list" }' nomefile
- **print items >> nomefile**
- E' possibile utilizzare anche l'operatore “<”

Redirezione in awk

- Esecuzione di comandi

```
lso:~>awk '{  
    print $1 > "names.unsorted";  
    command = "sort -r > names.sorted";  
    print $1 | command  
}' file
```

Riferimenti

- Capitolo 4,5 e 6 di [Bash Guide for Beginners]
- Effective Awk Programming