Il linguaggio *awk*

Awk è un linguaggio di programmazione progettato per il trattamento di informazioni testuali con operazioni di ricerca, controllo e manipolazione. Le azioni da eseguire sul testo sono scritte usando brevi pezzi di codice che utilizzano parole chiave, sintassi e strutture di controllo molto simili al linguaggio C.
Può servire per estrarre dati da file di grandi dimensioni, per formattare testi oppure per ottenere dati da utilizzare con altri comandi.
Awk deriva il suo nome dalle iniziali dei suoi creatori: Alfred V. Aho, Peter J. Weinberger, e Brian W. Kernighan.
Nella sua versione più recente e più estesa, realizzata nell'ambito del progetto GNU, il programma è indicato con il nome **gawk**.
Awk è un programma di confronto sulla base di modelli (*pattern matching*) e utilizza due file: il file dei dati, che contiene le righe da esaminare, e il programma dei comandi che contiene le istruzioni da eseguire.
Il file dei dati è esaminato record per record: il record è normalmente una riga del file di testo e comunque una sequenza di caratteri separata dai successivi tramite il carattere \n (*newline*), a meno che il programmatore specifichi un diverso separatore di record assegnandone il valore alla variabile RS.
Ogni record è formato da campi, intesi come un insieme di caratteri separato dai successivi dal carattere spazio o tabulazione, a meno che il programmatore specifichi un diverso separatore di campo assegnato alla variabile FS.
Record e campi sono automaticamente associati a speciali variabili: la variabile $0 è un record (una riga intera), $1 è il primo campo (la prima parola), $2 il secondo e così via.

I comandi del programma possono essere di tre tipi:

- i comandi iniziali, che hanno come parola iniziale BEGIN, indicano le assegnazioni o le operazioni da eseguire una volta sola all'inizio, prima di iniziare la scansione del file di dati;
- i comandi di controllo rappresentano le operazioni da applicare a ciascuna riga del file di dati;
- i comandi finali, che hanno come parola iniziale END, sono le operazioni da eseguire una volta sola quando viene raggiunta la fine del file di dati.

Ogni comando ha la seguente struttura generale:

/ modello / { istruzioni }

se la riga rispetta le caratteristiche del modello (racchiuso tra una coppia di caratteri /), vengono eseguite su di essa le istruzioni racchiuse tra le parentesi graffe e separate tra loro da punto e virgola.
Se manca il modello, le istruzioni sono comunque eseguite su tutte le righe.
Le regole del modello possono essere anche più di una, combinate con gli operatori AND (indicato con &&), OR (indicato con ||) e NOT (indicato con !).

L'esecuzione di un programma *awk* ha la seguente forma generale:

```
awk -f filecomandi filedati
```

Normalmente poi si vuole che l'output del programma *awk* generi un nuovo file di testo, come risultato del trattamento dei dati contenuti nel file di partenza. Quindi la forma generale del programma *awk* contiene anche la ridirezione dell'output:

```
awk -f filecomandi filedati > fileoutput
```

I comandi, anziché essere raggruppati in un file, possono anche comparire direttamente nella linea di comando di *awk*, delimitati da una coppia di apici. Questa modalità è usata nei casi in cui i comandi da eseguire siano in numero limitato.
Per esempio, la seguente esecuzione scrive sul video la prima e la terza parola di tutte le righe del file di partenza, scambiandole di posizione:

```
awk '{ print $3, $1 }' filedati
```

Mancando la specificazione del modello, le azioni sono eseguite su tutte le righe del file di dati.
Con il comando seguente, invece, l'istruzione *print* è eseguita solo sulle righe che contengono la sequenza di caratteri *12*.

```
awk '/12/ { print $3, $1 }' filedati
```

I comandi principali sono:

Assegnazione
variabile = valore

Output dei valori di variabili
print (variabili)

Output formattato
printf ("stringa di formattazione", variabili)

Strutture di controllo
if (condizione) comando1 **else** comando2
while (condizione) comandi
for (assegnazione iniziale; (condizione); assegnazione) comandi

La sintassi è del tutto simile a quella del linguaggio C. In particolare il segno **=** indica l'assegnazione, mentre il doppio uguale **==** indica il segno di uguaglianza nella scrittura delle condizioni. Inoltre la notazione **+=** indica l'incremento del valore di una variabile.

Alcune variabili predefinite, di uso comune nei programmi, sono:

NF	numero delle parole di una linea
NR	numero del record, cioé il numero della linea esaminata
FILENAME	nome del file di dati
FS	separatore di campi (normalmente spazio o tabulazione)
RS	separatore di record (normalmente *newline*)

Progetto 1

Dato il file vendite contenente su due colonne la data e l'importo, separate dalla tabulazione, calcolare la somma e la media degli importi del mese di dicembre.

Le azioni da eseguire sono scritte in un file di comandi di nome *progr1*, che contiene il seguente codice:

```
BEGIN { somma=0 }
  /dic/ { somma+=$2 }
END { print "Totale = ", somma, " Media = ", somma/NR }
```

La riga che inizia con *Begin* indica le azioni da compiere prima di esaminare le righe del file; la seconda riga esamina una ad una le righe del file ed esegue l'istruzione di incremento della somma solo per le righe che contengono la stringa *dic*. Infine la riga che inizia con *End* indica le operazioni da svolgere al raggiungimento della fine del file di dati. La variabile predefinita *NR* contiene il numero di riga del file, quindi alla fine del file restituisce il numero totale delle righe esaminate.

L'esecuzione deve essere poi avviata con il comando:

```
awk -f prog1 vendite
```

Le principali specifiche di formattazione per il comando **printf** sono:

- **%c** un singolo carattere
- **%d** un intero decimale
- **%e** numero a virgola mobile in notazione scientifica
- **%f** numero a virgola mobile in formato [-]ddd.ddd
- **%i** numero intero
- **%u** numero intero senza segno
- **%s** stringa
- **%%** scrive il simbolo "%".

Per esempio, utilizzando il comando *printf*, i risultati del precedente esempio possono essere visualizzati in modo formattato nel seguente modo:

```
END { printf "Totale = %6d Media = %6.2f \n", somma, somma/NR }
```

Le regole del modello per esaminare le righe del file di dati utilizzano le cosiddette espressioni regolari (*regular expression*), già viste con il comando *grep*, costruite usando i seguenti metacaratteri:

[ ] racchiudono un insieme di caratteri, ciascuno dei quali può comparire in quella posizione all'interno della stringa da cercare; se i caratteri sono in sequenza si può usare il segno - per indicare il range

. (punto) significa qualsiasi carattere in quella posizione

\ toglie il significato ai metacaratteri

^ indica la ricerca a partire dal primo carattere di ogni riga

$ indica la ricerca a partire dalla fine della riga.

Per esempio, il seguente comando *awk* estrae le righe di commento dal file *script1*, cioè le righe che hanno il carattere # nella prima posizione della riga.

```
awk '/^#/ {print $0}' script1
```

Lo stesso comando può essere applicato anche ad un insieme di file, usando la *file substitution*. Per esempio, il seguente comando estrae le righe di commento da tutti gli script aventi il nome che inizia con la lettera *u*:

```
awk '/^#/ {print $0}' u*
```

Progetto 2

Contare quante volte la parola "Europa" compare nelle righe del file testo1 (la parola può essere presente anche più volte in una stessa riga).

Il file di comandi di nome *progr2* contiene il seguente codice:

```
BEGIN { conta = 0 }
/Europa/ { for (i=1; (i <= NF); i+=1)
        { if ($i == "Europa") conta += 1 }}
END { print (" La parola Europa compare ", conta, " volte") }
```

L'esecuzione è attivata con il comando:

```
awk -f progr2 testo1
```

Progetto 3

Un file di indirizzi di nome indir contiene, per ciascuna riga, sei campi separati dalla virgola: cognome, nome, via o piazza, CAP, città, provincia. Il programma deve stampare gli indirizzi di Milano nella forma di etichette:
cognome nome
via
CAP città
(provincia)

Il file di comandi *etichette* contiene il seguente codice

```
BEGIN { FS="," }
/Milano/ { if ($5 == "Milano") then
         { printf " %s %s\n", $1, $2;
           printf "%s\n", $3;
           printf "%s %s\n", $4, $5;
           printf " (%s )\n", $6;
          printf "\n" }
         }
```

Prima di esaminare le righe, alla variabile *FS* è assegnato il carattere , (virgola) come nuovo separatore dei campi. Il programma contiene un doppio controllo sulla città "Milano", sia come modello, per estrarre le righe che contengono quella stringa, sia con una struttura *if* per essere certi che la stringa "Milano" indichi la città (il quinto campo, variabile $5); infatti un indirizzo potrebbe contenere la stringa "Milano" nell'indirizzo (per esempio, "via Milano" oppure "Piazza Milano").
L'esecuzione del programma viene lanciata con il comando:

```
awk -f etichette indir
```