



Espressioni Regolari

Roberto De Virgilio

Sistemi operativi - 15 Ottobre 2017

Espressioni regolari in Unix

- ◆ **Patterns (modelli)** che individuano insiemi di stringhe (linguaggi) e rappresentano uno strumento piuttosto potente e sofisticato di UNIX.
- ◆ In altre parole, è una stringa scritta secondo una precisa sintassi
- ◆ Utilizzate da vari programmi di elaborazione di testi (ad es. **grep**, **sed**, **awk**, **vi**, **emacs**) per:
 - elencare/eliminare righe che contengono un'espressione regolare
 - sostituire un'espressione regolare (-find/replace)
 - ...

Sintassi delle espressioni regolari

- ◆ **insiemi di caratteri:** pattern elementari che specificano la presenza un carattere appartenente ad un certo insieme.
- ◆ **ancore:** legano il pattern a comparire una posizione specifica della riga (es. inizio, fine).
- ◆ **gruppi:** “racchiudono” l'espressione regolare che può quindi essere riferita come una singola entità.
- ◆ **modificatori:** specificano ripetizioni dell'espressione che precede il modificatore stesso.

Sintassi delle espressioni regolari

- ◆ **insiemi di caratteri:** pattern elementari che specificano la presenza un carattere appartenente ad un certo insieme.
- ◆ **ancore:** legano il pattern a comparire una posizione specifica della riga (es. inizio, fine).
- ◆ **gruppi:** “racchiudono” l'espressione regolare che può quindi essere riferita come una singola entità.
- ◆ **modificatori:** specificano ripetizioni dell'espressione che precede il modificatore stesso.

Insieme di caratteri

- ◆ Per identificare diversi caratteri è possibile utilizzare le parentesi quadre `[...]`.
- ◆ Una lista di caratteri racchiusa tra parentesi quadre corrisponde a qualsiasi carattere in esse contenuto.
- ◆ Se la lista viene preceduta dal simbolo `^`, l'espressione corrisponde a qualsiasi carattere non contenuto nella lista.

Esempio: `[^abcd]`

fughe mostrata perché contiene f,u,g,h,e

coda mostrata perché contiene o

bada non mostrata perché non contiene caratteri diversi da a,b,c,d

Insieme di caratteri

- ◆ Il carattere \wedge , se specificato dopo il primo carattere, assume significato letterale
- ◆ L'espressione **[a \wedge b]** mostra le linee del file che contengono i caratteri **a**, \wedge e **b** e non caratteri **a** seguiti da caratteri diversi da **b**.
- ◆ All'interno delle parentesi quadre è possibile indicare un **intervallo** di caratteri specificando gli estremi e separandoli con il carattere **-**.
- ◆ L'espressione **[a-g]** mostra tutte le righe del file che contengono una lettera compresa tra **a** e **g**.
- ◆ Alcune localizzazioni prevedono l'ordinamento **aAbBcCdD...**, per cui specificando **[a-g]** si individuano le lettere **[aAbBcCdDeEfFg]**.

Insieme di caratteri

- ◆ *Si possono anche identificare classi di caratteri attraverso le espressioni:*

<code>[:alnum:]</code>	Caratteri alfanumerici (cifra o lettera)
<code>[:alpha:]</code>	Caratteri alfabetici
<code>[:digit:]</code>	Caratteri numerici
<code>[:lower:]</code>	Lettere minuscole
<code>[:upper:]</code>	Lettere maiuscole
<code>[:xdigit:]</code>	Cifre esadecimali

...

Esempio: `[:lower:]` Seleziona tutte le righe che contengono una minuscola.

Concatenazione

- ◆ Due espressioni regolari, **regex1** e **regex2**, possono essere concatenate in una nuova espressione regolare **regex1 regex2**.
- ◆ Ad essa corrisponde qualsiasi stringa formata da due stringhe concatenate corrispondenti alle singole espressioni di partenza.
- ◆ La più semplice forma di concatenazione è una stringa di lettere o numeri.

Concatenazione

- ◆ **Esempio1:** L'espressione regolare '**ab**' ricerca tutte le righe che contengono la stringa **ab**.
- ◆ **Esempio2:** L'espressione regolare '**a{1-5}**' restituisce tutte le righe contenenti una delle seguenti stringhe:
{a₁,a₂,a₃,a₄,a₅}.

Alternanza

- ◆ Due espressioni regolari, **regex1** e **regex2**, possono essere concatenate utilizzando l'operatore pipe **|**. Funziona solo nelle “extended regular expression”.
- ◆ L'espressione regolare risultante corrisponde a qualsiasi stringa che corrisponda alla prima **o** alla seconda.
- ◆ L'insieme delle stringhe rappresentate dall'espressione **regex1|regex2** è l'**unione** degli insiemi delle stringhe rappresentate da **regex1** e **regex2**.

Alternanza

- ◆ **Esempio:** L'espressione regolare '**a|b**' visualizza le righe che contengono le stringhe che contengono la lettera **a** e quelle che contengono la lettera **b**.

Posizionamento

- ◆ I caratteri `^` e `$` identificano all'interno di un'espressione regolare rispettivamente **inizio** e **fine riga**.
- ◆ **Esempio1:** L'espressione regolare `'stringa$'` stampa tutte le righe che **terminano con stringa**.
- ◆ **Esempio2:** L'espressione regolare `'^stringa'` stampa tutte le righe che **iniziano con stringa**.
- ◆ **Esempio3:** L'espressione regolare `'^stringa$'` stampa le righe costituite solo dalla parola **stringa**.

Posizionamento

- ◆ I simboli `\<` e `\>` individuano rispettivamente l'**inizio** e la **fine** di una **stringa**.
- ◆ **Esempio1:** L'espressione regolare '`\<stringa`' stampa tutte le righe che contengono una **parola** che **comincia** con **stringa** (ad es.: **stringa**, **stringato**, ma non **costringa**).
- ◆ **Esempio2:** L'espressione regolare '`stringa\>`' stampa la tutte le righe che contengono una **parola** che **termina** con **stringa** (ad es.: **stringa**, **costringa**, ma non **stringato**).

Caratteri speciali della shell ed espressioni regolari

- ✿ *Le espressioni regolari possono generare confusione perché utilizzano gli stessi caratteri speciali della shell, ma con significati a volte diversi.*
- ✿ *L'espansione dei caratteri speciali della shell viene sempre effettuata prima di eseguire un comando.*
- ✿ *Per preservare le espressioni regolari utilizzate da alcuni comandi (come ad esempio **grep**) da interpretazioni errate, è opportuno proteggerle con gli apici '...', evitando così che la shell interpreti i caratteri speciali di shell contenuti nell'espressione regolare. Questo è necessario anche quando i caratteri speciali hanno la stessa funzione.*

Caratteri speciali per le espressioni regolari

- ◆ *Il carattere punto . è un carattere speciale che rappresenta qualsiasi carattere (lettera, cifra, spazio, tab, ...).*
- ◆ *I caratteri speciali all'interno di un'espressione regolare con le quadre [...] perdono di significato e vengono considerati letteralmente.*
- ◆ **Esempio:** L'espressione regolare '\.\.*' mostra le linee del file che contengono i caratteri \, . e * e non zero o più occorrenze del carattere .

Operatori di ripetizione

- ◆ Un'espressione regolare che genera una corrispondenza con un singolo carattere può essere seguita da uno o più **operatori di ripetizione**.
- ◆ Il carattere * viene specificato per cercare un'espressione ripetuta un numero qualsiasi di volte (anche zero).
- ◆ **Esempio:** L'espressione regolare 'ab*c' visualizza le righe che contengono le stringhe {ac,abc,abbc,abbbc,ab...bc}.

Operatori di ripetizione

- ◆ Il carattere **?** viene specificato per cercare un'espressione ripetuta al più una volta (anche zero). Funziona solo nelle “extended regular expression”
- ◆ **Esempio:** L'espressione regolare '**ab?c**' visualizza le righe che contengono le stringhe **{ac,abc}**, ma non corrisponde alle stringhe **{abbc,abbbc,ab...bc}**.
- ◆ Il carattere **+** viene specificato per cercare un'espressione ripetuta almeno una volta. Funziona solo nelle “extended regular expression”
- ◆ **Esempio:** L'espressione regolare '**ab+c**' visualizza le righe che contengono le stringhe **{abc,abbc,abbbc,ab...bc}**, ma non corrisponde alla stringa **{ac}**.

Operatori di ripetizione

- ◆ L'espressione $\{n\}$ viene specificata per cercare un'espressione ripetuta esattamente n volte. Funziona solo nelle “extended regular expression”.
- ◆ **Esempio:** L'espressione 'ab{2}c' visualizza le righe che contengono le stringhe {abbc}, ma non corrisponde alle stringhe {ac,abc,abbbc,ab...bc}.
- ◆ L'espressione $\{n,m\}$ viene specificata per cercare un'espressione ripetuta **almeno n** volte e **al più m** volte. Funziona solo nelle “extended regular expression”.
- ◆ **Esempio:** L'espressione regolare 'ab{2,5}c' visualizza le righe che contengono le stringhe {abbc,abbbc,abbbb,abbbbb}, ma non corrisponde alle stringhe {ac,abc,abbb...bbc}.
- ◆ L'espressione $\{n,\}$ viene specificata per cercare un'espressione ripetuta **almeno n volte**. Funziona solo nelle “extended regular expression”.
- ◆ **Esempio:** L'espressione regolare '\<[:digit:]\>{4,}\>' visualizza le righe contenenti stringhe che cominciano con numeri aventi almeno quattro cifre.

Regole di precedenza

- ◆ *Nell'interpretazione delle espressioni regolari le operazioni sulle espressioni vengono valutate nel seguente ordine*
 1. ripetizione
 2. concatenazione
 3. alternanza
- ◆ *Le parentesi tonde possono essere utilizzate per modificare quest'ordine predefinito.*

Regole di precedenza

- ◆ Esempio: L'espressione regolare ' $a|bc^+$ ', esplicitando l'ordinamento delle operazioni, equivale a: ' $(a|(b(c^+)))$ ' ovvero ricerca stringhe del tipo $\{a, bc, bcc, bccc\dots\}$.
- ◆ Con le parentesi tonde possiamo modificare l'ordinamento delle operazioni:
 - con ' $a|(bc)^+$ ' ricerchiamo stringhe del tipo $\{a, bc, bcbc\dots\}$;
 - con ' $(a|b)c^+$ ' ricerchiamo stringhe del tipo $\{ac\dots, bc\dots\}$.

Uso di grep

- ◆ Il comando **grep** (*Global Regular Expression Print*) viene utilizzato per cercare stringhe all'interno di un file.
- ◆ **SINTASSI:** *grep [opzioni] 'pattern' [file]*, dove **pattern** è un'espressione regolare. Se il file non è specificato, grep opera sullo standard input.
- ◆ **Esempio:** proviamo a digitare
grep '^root:' /etc/passwd

Esercizi

Costruiamo il file di testo esempio.txt con il seguente

contenuto:

```
fughe
coda
bada
MAIUSCOLE
a1
a2
a3
a4
a5
a6
a7
a8
a9
stringa
stringato
costringa
ac
abc
abbc
abbbc
file.txt
ciao a *
exit
^ ^
a^
done
matr 543672
matr 543321
matr 523455
tel 0801234567
prefix 080
a
b
bc
bcc
bccc
bcbcbc
bacbc
bac
acc
caa
acac
```

Esercizi

- ◆ *Mostra tutte le linee del file che contengono la lettera a almeno una volta in qualsiasi punto*
 - ▶ *grep a esempio.txt*

Esercizi

- ◆ *Mostra tutte le linee del file che contengono almeno un carattere*
- ▶ ***grep . esempio.txt***

Esercizi

- ◆ *Mostra tutte le linee del file che contengono almeno una tra le lettere **a,b,c,d***
 - ▶ ***grep '[abcd]' esempio.txt***

Esercizi

- ◆ *Mostra le linee del file che contengono almeno una lettera diversa da a,b,c,d*
 - ▶ *grep ‘[^abcd]’ esempio.txt*

Esercizi

- ◆ *Mostra tutte le righe del file che contengono una lettera compresa tra **a** e **g***
 - ▶ ***grep '[a-g]' esempio.txt***

Esercizi

- ✿ *Stampa tutte le righe che contengono un carattere maiuscolo*
 - ▶ `grep [:upper:] esempio.txt`

Esercizi

- ◆ *Stampa tutte le linee del file che contengono la stringa ad*
 - ▶ *grep 'ad' esempio.txt*

Esercizi

- ◆ *Ricerca una stringa di due caratteri: il primo deve essere la lettera **a** e il secondo un numero compreso tra **1** e **5***
- ▶ ***grep 'a[1-5]' esempio.txt***

Esercizi

- ◆ *Stampa tutte le righe che terminano con la parola **stringa***
 - ▶ *grep ‘stringa\$’ esempio.txt*

Esercizi

- ◆ *Stampa tutte le righe che iniziano con la parola **stringa***
 - ▶ *grep '^stringa' esempio.txt*

Esercizi

- ◆ *Stampa tutte le righe che contengono soltanto la parola **stringa***
 - ▶ ***grep '^stringa\$' esempio.txt***

Esercizi

- ◆ *Stampa tutte le righe che contengono una parola che comincia con la parola **stringa***
 - ▶ *grep '\<stringa' esempio.txt*

Esercizi

- ◆ *Stampa tutte le righe che contengono una parola che termina con la parola **stringa***
 - ▶ `grep 'stringa\>' esempio.txt`

Esercizi

- ◆ Visualizza le righe che contengono le stringhe del tipo: *ac*, *abc*, *abbc*, *abbbc*, *ab...bc*
 - ▶ ***grep 'ab*c' esempio.txt***

Esercizi

- ◆ Visualizza le righe che contengono i caratteri: ., \, e * e non zero o più occorrenze del carattere.
 - ▶ ***grep '\.*' esempio.txt***

Esercizi

- ◆ Visualizza le righe che contengono i caratteri: *a*, *^*, e *b*.
- ▶ *grep '[a^b]' esempio.txt*

Esercizi

- ◆ Visualizza le righe che contengono le stringhe: *ac*, *abc*, ma non le stringhe che contengono più di una *b* tra *a* e *c*.
 - ▶ ***grep -E 'ab?c' esempio.txt***

Esercizi

- ◆ Visualizza le righe che contengono le stringhe: *abc*, *abbc*, *abbbc*, *abb..bbc* ma non le stringhe che contengono *ac*.
 - ▶ ***grep -E 'ab+c' esempio.txt***

Esercizi

- ✿ Visualizza le righe che contengono le stringhe: *abbc*.
 - ▶ ***grep -E 'ab{2}c' esempio.txt***

Esercizi

- ◆ Visualizza le righe che contengono le stringhe: *abbc* e *abbbc*.
- ***grep -E 'ab{2,3}c' esempio.txt***

Esercizi

- ◆ Visualizza le righe che contengono numeri con almeno **4 cifre**.
 - ▶ ***grep -E '\<{[:digit:]}{4,}\>' esempio.txt***

Esercizi

- ◆ Visualizza le righe che contengono la lettera *a* o la lettera *b*.
 - ▶ ***grep -E 'a|b' esempio.txt***

Esercizi

- ◆ *Visualizza le righe che contengono la lettera a o stringhe che iniziano con la lettera b seguita da almeno una lettera c (bc, bcc, bccc, ...)*
 - ▶ **grep -E 'a|bc+' esempio.txt**

Esercizi

- ◆ *Visualizza le righe che contengono la lettera a o almeno una occorrenza della stringa bc(bc, bc₂c, bc₃c₂c, ...)*
 - ▶ ***grep -E 'a|(bc)+' esempio.txt***

Esercizi

- ◆ Visualizza le righe che contengono stringhe del tipo *ac*, *acc*, *acc**c*, ... o stringhe del tipo *bc*, *bcc*, *bccc*, ...
 - ▶ ***grep -E '(a|b)c+'* esempio.txt**

Esercizi

- ◆ *Visualizza le righe che contengono parole che iniziano con la lettera a e non finiscono con una cifra pari*
 - ▶ `grep '\<a[^02468]\>' esempio.txt`

