

# Espressioni regolari

# **grep** [*opzioni*] *pattern* [*nomefile*]

- Stampa le righe del file che corrispondono al pattern
- Il pattern è una *espressione regolare*
  - nel caso più semplice, il pattern può essere una stringa senza caratteri speciali
  - `grep a pippo.txt` stampa le righe di `pippo.txt` che contengono una `a`
- Se *nomefile* non è specificato, legge da standard input
  - `ls -l | grep 2006`      elenca i file che sono stati modificati l'ultima volta nel 2006 (oltre a...)
  - `ls -l | grep rwx`      elenca i file per cui almeno una categoria di utenti ha tutti i permessi (oltre a...)
- Con l'opzione “-v”, stampa le righe che *non* corrispondono al pattern

# Basic Regular Expressions

- `.` qualunque carattere (1)
- `exp*` zero o più occorrenze di `exp` (2)
- `^exp` `exp` all'inizio del rigo (1)
- `exp$` `exp` alla fine del rigo (1)
- `[a-z]` un carattere nell'intervallo specificato
- `[^a-z]` un carattere fuori dall'intervallo

Note: (1) è un carattere normale per Bash  
(2) ha un significato diverso per Bash

# Basic Regular Expressions

- `\<exp`      `exp` all'inizio di una parola      (1)
- `exp\>`      `exp` alla fine di una parola      (1)
  
- `\t, \n`      tabulazione e fine riga

Note: (1) è un carattere normale per Bash  
(2) ha un significato diverso per Bash

# Esempi

- `a*b` zero o più a seguite da una b
- `a.*b` una a prima di una b
- `\<[[:upper:]]` una parola che inizia con lettera maiuscola
- `^d` la lettera d all'inizio del rigo
- `^a*$` un rigo vuoto o composto solo di a
- `^a.*b$` un rigo che inizia con a e finisce con b
- `\<.-` una parola che ha un trattino al secondo posto

# Extended Regular Expressions

- `exp+` una o più occorrenze di `exp` (1)
  - `exp?` zero o una occorrenza di `exp` (2)
  - `exp1 | exp2` `exp1` oppure `exp2` (2)
  - `( exp )` equivalente a `exp`, serve per stabilire l'ordine di valutazione
- 
- In **grep**, questi simboli vanno preceduti da “\” (backslash)
  - In **egrep** (*extended grep*), vanno usati direttamente

Note: (1) è un carattere normale per Bash  
(2) ha un significato diverso per Bash

# Esempi

- `[[[:digit:]]\+` una sequenza non vuota di cifre
- `^a\|b` un rigo che inizia con a oppure contiene b
- `^\(a\|b\)` un rigo che inizia con a oppure con b
- `\(\.txt\)\|\(\.doc\)\>` una parola che termina con .txt o con .doc  
(in egrep, `(\.txt)|\.doc` )

# Esercizi

- 1) Elencare i file con permesso di esecuzione per il proprietario
- 2) Elencare i file puntati da più di un hard link (\*)
- 3) Elencare le directory il cui nome inizia per maiuscola
- 4) Elencare i file con permesso di esecuzione oppure di scrittura per il gruppo di appartenenza

# Riferimenti

- Capitolo 4 di [Bash Guide for Beginners]