Gestione avanzata dei file

Espressioni jolly

- * qualunque insieme di caratteri
- ? un solo carattere qualunque
- ^ nega l'espressione seguente
- [ ] carattere singolo in un range di caratteri possibili
  - [a-c] le lettere a, b, c
  - [a-zA-Z] tutte le lettere maiuscole e minuscole

Comandi avanzati per la gestione dei file

- *file* permette di vedere il tipo di file
- *cut* “taglia” un file per colonne
- *paste* unisce le colonne di uno o più file
- *join* un paste più potente
- *split* divide i file
- *grep* cerca un espressione in un file
- *sort* per ordinare un file

Comandi avanzati per la gestione dei file

- *head* mostra le prime righe di un file
- *tail* mostra le ultime righe di un file
- *wc* conteggia lettere, parole e byte
- *diff* compara due file o due cartelle
- *cmp* compara due file
- *od* visualizzatore esadecimale
- *sed* editor di flusso
- *awk* linguaggio per elaborazione testi

head, tail, split

- *#head -n num file*
- *#tail -n num file*
  - Numero righe di default 10
  - Se si usa un numero negativo?
- *#split opzioni file*
  - -a numero di caratteri per il suffisso
  - -b dimensione massima in byte (KB, MB)
  - -l dimensione massima in linee di testo

sort

- *sort opzioni file*
  - -c verifica l'ordinamento
  - -o file outpu su file
  - -u elimina le linee duplicate
  - -f non distingue maiuscole e minuscole
  - -r ordinamento inverso
  - -n (-g) ordine numerico
  - -k chiavi per l'ordinamento
  - -b ignora gli spazi bianchi
  - -t delimitatore

wc

- *wc* conta righe, caratteri, parole (stringhe separate da uno o più spazi) e byte di un file
  - -l conta le linee
  - -w conta le parole
  - -c conta i byte
  - -m conta i caratteri
- L'opzione di default è -lwc

cut, paste

- *cut*
  - -b byte
  - -c caratteri
  - -f campi
    - Es.:
      - *#cut -c -3,8-10,40,42,50- nomefile*
  - -d specifica il delimitatore

- *paste*
  - -d specifica il delimitatore
    - Es:
      - *#paste -d " " file1 file2*

join

- *#join [-1 campo1] [-2 campo2] file1 file2*
  - -a filenumber aggiunge anche le linee non corrispondenti
  - -v filenumber solo le linee non corrispondenti del file filenumber
  - -1 campo1 specifica il campo da considerare per file1
  - -2 campo2 specifica il campo per il file2
  - -o list specifica i campi da stampare
  - -t delimitatore specifica il delimitatore

  - Es: *#join -t “;“ -1 2 -2 3 file1 file2*

- N.B.: i campi da collegare devono essere ordinati con *sort -b*

diff, cmp

- *#cmp opzioni file1 file2*
  - -l elenca le differenze con posizione (in byte) e valori dei due byte nei rispettivi file.
  - -s output silente in caso di differenze
- *#diff opzioni file1 file2*
  - -b (-w) ignora gli spazi bianchi
  - -i ignora maiscuole, minuscole
  - -r ricorsivo (per confronto tra cartelle)
  - -s non riporta i nomi di file differenti

sed, awk

- Qui trattiamo solo un paio di esempi per ciascuno di questi due comandi/linguaggi
  - *#awk '{ if($7<0.5) print NR,$5,$6}' ppp*
    - Stampa, dal file "ppp", un intero progressivo, le colonne $5 e $6, ma solo dalle righe che verificano la condizione: colonna 7<0.5
  - *#sed s/testo1/testo2/g nomefile*
    - Sostituisce la stringa testo1 con la stringa testo2 in tutto il file nomefile

od

- Octal Dump permette di vedere anche file con caratteri non ASCII, come file binari.
  - -c output in caratteri
  - -x output in esadecimale
    - *#od -xc /sbin/ifconfig | head*

```
bash-3.00# od -xc /sbin/ifconfig | head
0000000    457f    464c    0101    0001    0000    0000    0000    0000
         177   E   L   F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0
0000020    0002    0003    0001    0000    2328    0805    0034    0000
         002  \0 003  \0 001  \0  \0  \0   (   # 005  \b   4  \0  \0  \0
0000040    e540    0000    0000    0000    0034    0020    0006    0028
           @   å  \0  \0  \0  \0  \0  \0   4  \0      \0 006  \0   (  \0
0000060    0017    0015    0006    0000    0034    0000    0034    0805
         027  \0 025  \0 006  \0  \0  \0   4  \0  \0  \0   4  \0 005  \b
0000100    0000    0000    00c0    0000    00c0    0000    0005    0000
          \0  \0  \0  \0   À  \0  \0  \0   À  \0  \0  \0 005  \0  \0  \0
```

grep

- *egrep* = *grep -E* grep con espressioni regolari
- *fgrep* = *grep -F* grep “rapido”
- Opzioni
  - -c solo conteggio righe output
  - -n scrive il numero di riga
  - -v nega la corrispondenza
  - -x solo righe che corrispondono interamente
  - -l solo i nomi dei file
  - -i ignora maiuscole/minuscole
  - -h non scrive il nome del file

grep

- Es.:
  - *grep root /etc/group*
    - Cerca root nel file dei gruppi
  - *grep -v root /etc/group*
    - Cerca le linee senza root nel file dei gruppi
  - *grep -n root /etc/passwd*
    - Riporta anche il numero di riga dove ha trovato “root”

Espressioni regolari (opzionale)

- Un'espressione regolare è un modello che descrive un insieme di stringhe
- Purtroppo, nonstante le specifiche POSIX ci sono molte implementazioni.
  - . un solo carattere qualunque
  - ^ inizio di una riga
  - $ fine di una riga
  - [ ] elenco di caratteri
    - [aeiou] trova una vocale
    - [0-9] trova un numero tra 0 e 9
    - [0-9a-z] trova un numero o un carattere minuscolo
    - ^ nega l'elenco

Espressioni regolari (opzionale)

- \< inizio di una “parola”
- \> fine di una “parola”
- ? estende la ricerca anche a 0 occorrenze
- * estende la ricerca a molte occorrenze
  - Esempi:
    - a.?c ac, abc, acc corrispondono, ma non abec
    - a.c abc, acc, ma non ac
    - a*c ac, aac, aaac, aaaaac
- {n} estende la ricerca a n occorrenze esatte