Ulteriori comandi sui file

- Confronto tra file:

  1. `> cmp file1 file2`
     restituisce il primo byte ed il numero di linea in cui `file1` e `file2` differiscono (se sono uguali, non viene stampato nulla a video).
  2. `> diff file1 file2`
     restituisce la lista di cambiamenti da apportare a `file1` per renderlo uguale a `file2`.

- Ricerca di file:
  `> find` <*pathnames*> <*expression*>
  attraversa ricorsivamente le directory specificate in <*pathnames*> applicando le regole specificate in <*expression*> a tutti i file e sottodirectory trovati.
  <*expression*> può essere una fra le seguenti:
  1. opzione,
  2. condizione,
  3. azione.

Esempi d'uso di `find`

- `> find . -name '*.c' -print`
  cerca ricorsivamente a partire dalla directory corrente tutti i file con estensione `c` e li stampa a video.

- `> find . -name '*.bak' -ls -exec rm {} \;`
  cerca ricorsivamente a partire dalla directory corrente tutti i file con estensione `bak`, li stampa a video con i relativi attributi (`-ls`) e li cancella (`-exec rm {} \;` Il carattere `\` serve per fare il "quote" del `;`).

- `> find /etc -type d -print`
  cerca ricorsivamente a partire dalla directory `/etc` tutte e solo le sottodirectory, stampandole a video.

I comandi filtro

I **filtri** sono una particolare classe di comandi che possiedono i seguenti requisiti:

- prendono l'input dallo **standard input device**,
- effettuano delle operazioni sull'input ricevuto,
- inviano il risultato delle operazioni allo **standard output device**.

Tali comandi risultano quindi degli ottimi strumenti per costruire pipeline che svolgano compiti complessi.

Ad esempio:

```
> uniq file
```

restituisce in output il contenuto del file `file`, sostituendo le linee adiacenti uguali con un'unica linea.

Comandi filtro: `grep`, `fgrep`, `egrep`

I comandi:

- `grep`: General Regular Expression Parser,
- `fgrep`: Fixed General Regular Expression Parser,
- `egrep`: Extended General Regular Expression Parser,

restistuiscono solo le linee dell'input fornito che contengono un **pattern** specificato tamite espressione regolare o stringa fissata.

Sintassi:

`grep` [options] pattern [filename...]
`fgrep` [options] string [filename...]
`egrep` [options] pattern [filename...]

Opzioni:

`-i`: ignora la distinzione fra lettere maiuscole e minuscole,
`-l`: fornisce la lista dei file che contengono il pattern/string,
`-n`: le linee in output sono precedute dal numero di linea,
`-v`: vengono restituite solo le linee che **non** contengono il pattern/string,
`-w`: vengono restituite solo le linee che contengono il pattern/string come parola completa,
`-x`: vengono restituite solo le linee che coincidono esattamente con pattern/string.

I metacaratteri delle espressioni regolari

metacarattere	tipo	significato
`^`	B	inizio della linea
`$`	B	fine della linea
`\<`	B	inzio di una parola
`\>`	B	fine di una parola
`.`	B	un singolo carattere (qualsiasi)
`[str]`	B	un qualunque carattere in `str`
`[^str]`	B	un qualunque carattere non in `str`
`[a-z]`	B	un qualunque carattere tra `a` e `z`
`\`	B	inibisce l'interpretazione del metacarattere che segue
`*`	B	zero o più ripetizioni dell'elemento precedente
`+`	E	una o più ripetizioni dell'elemento precedente
`?`	E	zero od una ripetizione dell'elemento precedente
`{j,k}`	E	un numero di ripetizioni compreso tra `j` e `k` dell'elemento precedente
`s|t`	E	l'elemento `s` oppure l'elemento `t`
`(exp)`	E	raggruppamento di `exp` come singolo elemento

dove B (basic) indica che la sequenza di caratteri è utilizzabile sia in `grep` che in `egrep`, mentre E (extended) indica che la sequenza di caratteri è utilizzabile solo in `egrep` (o in `grep` usando l'opzione `-E`).

Esempi d'uso di `grep`, `fgrep`, `egrep`

- `> fgrep rossi /etc/passwd`
  fornisce in output le linee del file `/etc/passwd` che contengono la stringa **fissata** `rossi`.

- `> egrep -nv '[agt]+' relazione.txt`
  fornisce in output le linee del file `relazione.txt` che **non** contengono stringhe composte dai caratteri `a`, `g`, `t` (ogni linea è preceduta dal suo numero).

- `> grep -w print *.c`
  fornisce in output le linee di tutti i file con estensione `c` che contengono la parola **intera** `print`.

- `> ls -al . | grep '^d.......w.'`
  fornisce in output le sottodirectory della directory corrente che sono modificabili dagli utenti ordinari.

- `> egrep '[a-c]+z' doc.txt`
  fornisce in output le linee del file `doc.txt` che contengono una stringa che ha un prefisso di lunghezza non nulla, costituito solo da lettere `a`, `b`, `c`, seguito da una `z`.

Comandi filtro: `sort`

Il comando `sort` prende in input delle linee di testo, le **ordina** (tenendo conto delle opzioni specificate dall'utente) e le invia in output.

- `sort` tratta ogni linea come una collezione di vari campi separati da delimitatori (default: spazi, tab ecc.).

- l'ordinamento di default avviene in base al **primo** campo ed è **alfabetico**.

Il comportamento di default si può cambiare tramite le opzioni:

`-b`	ignora eventuali spazi presenti nelle chiavi di ordinamento,
`-f`	ignora le distinzioni fra lettere maiuscole e minuscole,
`-n`	considera numerica (invece che testuale) la chiave di ordinamento
`-r`	ordina in modo decrescente,
`-o` *file*	invia l'output al file *file* invece che allo standard output,
`-t` *s*	usa *s* come separatore di campo,
`-k` *s1*`,`*s2*	usa i campi da *s1* a *s2*-1 come chiavi di ordinamento.

Esempi d'uso di `sort`

Volendo ordinare le righe del file `/etc/passwd` in base al terzo campo (user ID), il comando

```
> sort -t: -k3,4 /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
guest:x:1001:120:Guest User:/home/guest:/usr/local/bin/bash
daemon:x:1:1::/:
...
```

non dà il risultato voluto in quanto di default l'ordinamento è alfabetico, mentre il campo user ID è un numero; quindi si rende necessaria l'opzione `-n`:

```
> sort -t: -k3,4 -n /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
...
guest:x:1001:120:Guest User:/home/guest:/usr/local/bin/bash
...
```

Si noti che in entrambi gli esempi il separatore (`:`) è stato impostato con l'opzione `-t:`.

Comandi filtro: `tr`

Character translation: `tr` è un semplice comando che permette di eseguire operazioni come la conversione di lettere minuscole in maiuscole, cancellazione della punteggiatura ecc. Siccome può prendere input soltanto dallo standard input e stampare soltanto sullo standard output, bisogna usare delle pipe o delle ridirezioni di input/output per farlo leggere/scrivere su file.

Sintassi di base: `> tr str1 str2`
(i caratteri in `str1` vengono sostituiti con i caratteri in posizione corrispondente della stringa `str2`)

Esempi:

- `> tr a-z A-Z`
  converte le minuscole in maiuscole.

- `> tr -c A-Za-z0-9 ' '`
  sostituisce tutti i caratteri **non** (opzione `-c`: complemento) alfanumerici con degli spazi.

- `> tr -cs A-Za-z0-9 ' '`
  come nell'esempio precedente, ma comprime gli spazi adiacenti in un'unico spazio (opzione `-s`: *squeeze*).

- `> tr -d str`
  cancella i caratteri contenuti nella stringa `str`.

Cut and paste

- Il comando `cut` serve ad estrarre delle colonne specifiche dalle linee di testo che riceve in input:

```
> cut -d: -f1 /etc/passwd
root
daemon
bin
...
```

  il separatore si specifica con l'opzione `-d` (*delimiter*), il campo da estrarre con l'opzione `-f` (*field*).

- Il comando `paste` combina le righe corrispondenti di due file, inserendo un delimitatore fra esse (default: `<Tab>`):

```
> cd; cut -d: -f1 /etc/passwd > p1.txt; cut -d: -f6 /etc/passwd > p6.txt
> paste p1.txt p6.txt
root    /
daemon  /
bin     /usr/bin
...
```

Esercizi

- Qual è l'effetto del comando `sort file >file`, dove `file` è il nome di un file?

- Fare alcuni esperimenti per scoprire qual è l'effetto del comando `tr str1 str2` se le stringhe `str1` e `str2` hanno lunghezze diverse.

- Scrivere un comando per sostituire tutti i caratteri alfanumerici nell'input con un carattere <`Tab`>, in modo che non compaiano più <`Tab`> consecutivi.

- Il comando `date` fornisce data e ora su standard output. Scrivere una pipeline per estrarre soltanto i minuti.

- Scrivere una pipeline che permetta di scoprire se ci sono linee ripetute in un file.

- Visualizzare su standard output, senza ripetizioni, lo user ID di tutti gli utenti che hanno almeno un processo attivo nel sistema.