Corso di Laurea di Primo Livello
Scuola Universitaria Interfacoltà
in Biotecnologie
Università degli Studi di Torino

Corso di Abilità Informatiche
Secondo Modulo

AA 2008/2009

LABORATORIO INFORMATICO, LEZIONE 7:

ESPRESSIONI REGOLARI (REGULAR EXPRESSIONS)

PREAMBOLO

Vista la sua importanza, la 7ª lezione del laboratorio riprende e approfondisce l'argomento delle espressioni regolari (regular expressions). Le espressioni regolari possono essere molto complesse e anche molto difficili da imparare, ma sono uno strumento potente per molti scopi. Esiste una pagina `man` che li spiega in modo più dettagliato: `man 7 regex` (nella sezione 7 del manuale online; non "`man regex`" che invece spiega il funzionamento di particolari funzioni del linguaggio di programmazione C che trattano le espressioni regolari).

Prima di iniziare gli erercizi create una cartella `abInfo2_lab07` in `/home/studente`. Tutti gli esercizi vanno svolti in quella cartella, che alla fine della lezione deve essere cancellata con tutto il suo contenuto.

ESERCIZIO 1: I metacaratteri per specificare nomi di file

Esistono diverse sintassi simili alle espressioni regolari, cioè la sintassi precisa con cui va specificato un pattern può variare leggermente tra un'applicazione e l'altra, anche se il principio di base è lo stesso. Riprendiamo la sintassi più semplice che conoscete già: quella usata per specificare più di un file sulla command line (i *metacaratteri*). La seguente lista (incompleta) descrive alcune particolarità di questa sintassi (paragonatele con le specifiche analoghe descritte nel secondo esercizio!):

- simbolo per specificare esattamente un carattere qualsiasi: `?`
- simbolo per specificare un numero arbitrario (anche zero) di caratteri qualsiasi: `*`
- simbolo per specificare una carattere tra una scelta precisa: esempi: `[a-e]`, `[0-3aA]`

Fate i seguenti esecizi senza spostarvi dalla cartella home dello studente e con una sola riga di comando:

1) Contate i file in `/lib` il cui nome inizia con una "l".

2) Contate i file che si trovano in una delle sottocartelle di `/etc` con nome (della sottocartella) che inizia per "i".

3) Come nell'esercizio 2, ma i nomi dei file devono contenere almeno una cifra.

4) Contate i file in `/lib` che nel loro nome contengono una sequenza composta da una cifra seguita da un qualsiasi carattere, seguito da un'altra cifra (ad esempio "`5v7`").

ESERCIZIO 2: Espressioni regolari per il filtro `grep`

Uno dei filtri più utilizzati è `grep`. Riprendiamo gli esercizi della settimana scorsa. Scaricate l'archivio (file compattato) che avete usato la settimana scorsa (materiale del laboratorio 6) e scompattatelo usando il tool `tar`:

```
tar xvzf nome_del_file_archivio
```

(provate per curiosità anche a scoprire a cosa servono i parametri `x`, `v`, `z` e `f` del programma `tar`).

Ricordatevi che il comando grep si usa nel seguente modo:

```
grep [opzioni] 'pattern' nome_dei_file
```

dove il pattern può essere una parola o frase (mettendolo tra virgolette semplici) per ricercare le righe all'interno dei file che contengono esattamente questa parola o frase, oppure un'espressione regolare che costituisce un pattern per ricercare tutte le righe che soddisfano quel pattern. La pagina `man` vi informa su quali siano le possibili opzioni del comando (viste anche nell'ultimo laboratorio).

La sintassi delle *regex* usate per il comando `grep` è diversa dalla sintassi usata per specificare più di un file sulla command line (i *metacaratteri; vedete esercizio 1*). Ecco una lista (incompleta) di differenze:

- simbolo per specificare esattamente un carattere qualsiasi: **.**
- simbolo per specificare un numero arbitrario (anche zero) di caratteri qualsiasi: `.*`
  (notate la differenza: per i metacaratteri un asterisco solo indica un numero arbitrario di caratteri qualsiasi, mentre per `grep` *l'asterisco indica un numero arbitrario del carattere, o dei caratteri se in parentesi quadre, specificato prima*; in questo caso il punto indica un carattere qualsiasi, l'astersico indica che questo può comparire un numero arbitrario di volte).
- simbolo per specificare una sequenza di almeno un carattere qualsiasi: `.+`
  (il `+` indica che il carattere precedente deve comparire almeno una volta)

Prima di inizare gli esercizi con grep, leggete attentamente le slide (da pagina 8 a pagina 12) della sesta lezione che spiegano come si costruisce un'espressione regolare!

Importante: alcuni caratteri speciali delle espressioni regolari (se scritte sulla riga di comando) devono essere preceduti da un backslash (`\`), altrimenti la bash li interpreta in un altro modo. Esempi:

a) per specificare un numero di ripetizioni mediante le graffe (`{` e `}`) sulla command line bisogna scrivere: `\{n,m\}` (con `n`=numero minimo di ripetizioni; `m`=numero massimo).

b) per scegliere tra due o più opzioni si usa la pipe (`|`), che però per la bash ha il significato di concatenazioni di più comandi. Perciò sulla riga di comando quando usata per un'espressione regolare viene preceduta da un backslash. Esempio:
`protein\|aminoacid\|enzyme` invece di `protein|aminoacid|enzyme`.

c) per limitare la portata di una scelta (effetuata con `|`) si possono usare le parentesti tonde, che

però sulla riga di comando vanno precedute dal backslash. Esempio:
`\(AGCTTA\)*` invece di `(AGCTTA)*`.
d) per specificare che un carattere o una sottoespressione può comparire al massimo una volta si usa il punto interrogativo (`?`), che però per la command line ha un'altro significato (vedere l'esercizio 1 sui metacaratteri), e quindi va preceduto da un backslash. Esempio:
`AGCTA\?TA` invece di `AGCTA?TA`.
e) il simbolo `+` per specificare almeno un'occorenza di un carattere o sottocarattere va preceduta da un backslash. Esempio: `A\+` invece di `A+`.

1) Leggere un'espressione regolare può essere più facile che scriverla. Per prendere un pò di confidenza provate a leggere le seguenti espressioni regolari e individuate, tra la lista di stringhe sottostanti, quelle che soddisfano le condizioni dei pattern. Prendetevi un po di tempo, è facile che vi sfugga qualcosa (e alcuni dei pattern sono fatti appositamente per confondervi le idee ... :o)

   Pattern: (*attenzione*, per maggiore leggibilità i caratteri speciali interpretati diversamente dalla bash non sono preceduti dal backslash!)

I) `^t`	II) `[^t]`	III) `^[^t]*$`
IV) `[0-9]{3,5}`	V) `protein`	VI) `\<protein\>`
VII) `AA*`	VIII) `^[^p53]*$`	IX) `^[a-z0-9]*$`
X) `N|Ra` XI) `(N|R)a`	XII) `[0-9\.]{2}$`	XIII) `[Rnc]att?`
XIV) `CAA38095?1`	XV) `[A-Z]a(t+[^t]*){2,}`	XVI) `([aeiou].){2,}$`

   Stringhe:

a) Natale e' alle porte.	b) Troppa pubblicita' per giocattoli in tv.
c) tumor protein p53	d) protein p53
e) CAA38095.1	f) Rattus norvegicus
g) AF294441S1	h) Rat mRNA for nuclear oncoprotein p53
i) >gi|56828|emb|X13058.1	j) AAAAAACCAATGATCAAGAAAGTGGG
k) AF190269	l) alternatively spliced
m) gi|7243326|gb|AF190269.1	n) AF190269 Mus musculus p53 tumor suppr.
o) the Trp53 gene for transformation	p) DNA sequence from clone RP23-56I20

Eseguite i seguenti esercizi utilizzando il comando `grep` (con le opzioni che offre) e le espressioni regolari corrette per raggiungere lo scopo. Per vedere il contenuto dei file e farvi un idea di come sono fatti, usate uno dei tool che conoscete (sempre dalla bash, non usando l'interfaccia grafica).

2) Il file `mouse_sequences_nucleo.txt` contiene una lista di sequence nucleotidiche in formato FASTA (con una riga detta “header” che precede ogni sequenza e può contenere informazioni come l'identificatore, il nome e la descrizione della sequenza; le sequenze stesse possono avere più di una riga). Usate `grep` per:
   a) contare il numero di sequenze nel file.
   b) estrarre i soli header che contengono una C.
   c) estrarre solo le sequenze e contare il numero massimo di base nucleotidiche in una riga (vi serve anche un'altro comando per farlo, quale?)

3) Lavoriamo su tutti file che contengono sequenze (e solo su questi!). Usate `grep` per:
   a) elencare i file che contengono sequenze relative al suppressore tumorale p53.
   b) contare quante sequenze per p53 ci sono in ciascun file.
   c) contare quante sequenze per p53 ci sono in totale.
   d) estrarre tutti i header dei file che si riferiscono a sequenze relative ad altri geni o altre proteine. Quante sequenze sono?

4) Codoni di stop: TAG, TGA e TAA. Non usando il file dei pattern (`stop_codon.txt`), come nel laboratorio 6, ma espressioni regolari da specificare sulla riga di comando, provate a:

a) estrarre tutte le occorrenze di possibili codoni di stop dal file di sequenze nucleotidiche `homo_sequences_nucleo.txt`, facendo sì che venga restituito il numero di ogni riga ed evidenziato il codone di stop all'interno di ogni riga.
b) come 4a, ma: estrarre solo le occorrenze dei codoni di stop che non siano TAG, e con un'espressione regolare ridotta al minimo (4 lettere + simboli speciali).
c) estrarre tutti i header (da tutti i file di sequenze) che non contengono le ciffre 2,4 e 7. Quante sequenze sono e di che specie?
d) estrarre tutti i header (da tutti i file di sequenze) che sontengono la parola “protein”, ma *non* come parola a se stante (cioè all'interno di un'altra parola). Quante sequenze sono e di che specie?
e) estrarre (da tutti i file di sequenze) tutte le occorrenze almeno triple (cioè almeno tre volte una dopo l'altra) di un'A seguita da almeno una C. Fate sì che vi venga restituito (oltre al nome del file) il numero della riga e evidenziata l'occorenza del pattern di ricerca.

Alla fine di tutti gli esercizi, ricordatevi di cancellare la cartella `abInfo2_lab07` con tutto il suo contenuto (possibilmente con un singolo comando).