Espansione della shell

1) Espansione delle parentesi graffe

   p{a,e,i,o,u}zza

2) Espansione della tilde

   ~oliva

3) Espansione dei parametri e delle variabili

   $USER,$0

4) Sostituzione di comando

   $(wc -l < file)

5) Espansione aritmetica

   $(($i-1))

6) Suddivisione in parole

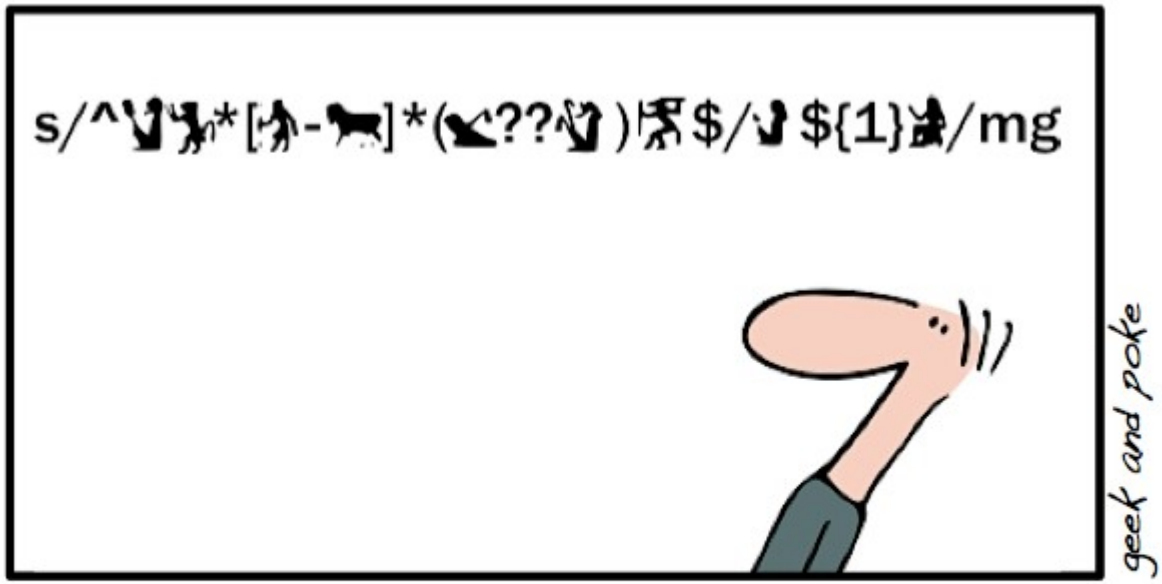
7) File globbing

   * ? [ ]

Questionario q1

- All'interno della vostra home directory create la directory copiabash e copiateci tutti i file il cui nome inizi per ".bash"
- Copiate il file .bash_history nel file copiahistory e modificatene i permessi in modo che non possa essere più visualizzato ma che possa essere cancellato
- Concatenate i file /etc/passwd e /etc/shadow reindirizzando l'output sul file account-out e lo standard error sul file account-err
- Contate i file contenuti nelle directory /usr/bin e /var
- Visualizzate il solo file che occupa più spazio di memoria nella vostra home directory
- Visualizzate la penultima riga del file /etc/passwd
- Create nella vostra home directory i file vuoti pazza, pezza, pizza, pozza, puzza, utilizzando l'espansione delle {} e della ~
- Utilizzate il comando echo per visualizzare la frase "2 * 3 > 5 espressione vera" utilizzando il solo carattere \ di escape

Laboratorio di sistemi operativi
A.A. 2010/2011
Gruppo 2
Gennaro Oliva
6
Espressioni Regolari



Espressioni Regolari

- Un'espressione regolare è un stringa che rappresenta un insieme di stringhe, scritta secondo una precisa sintassi
- Le espressioni regolari vengono tipicamente utilizzate nella ricerca di stringhe all'interno di file di testo e nella sostituzione delle stringhe trovate con nuovo testo
- Usando espressioni regolari nell'effettuare ricerche, non siamo obbligati a specificare esattamente i termini da ricercare, ma possiamo definire delle regole generali da utilizzare per trovare le stringhe che ci interessano

Esempi di possibili ricerche

- Esempi di ricerche e sostituzioni effettuabili con le espressioni regolari:
  - cerca una parola con 3 vocali
  - cerca una stringa che termina con 2 cifre qualsiasi
  - cerca una stringa con la prima lettera maiuscola
  - cerca una stringa che inizia con N86 e che termina con : e sostituisci eventuali zeri tra 86 e il primo numero diverso da `0` con il carattere '/'

Programmi

- Vari programmi utilizzano le espressioni regolari:
  - vi
  - less
  - sed
  - grep/egrep
  - awk
  - mutt
  - procmail
  - ...

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 il comando per cui per preservare le espressioni regolari da interpretazioni errate è opportuno proteggerle con gli apici

grep

- Il comando grep viene utilizzato per cercare stringhe all'interno di un file usando la seguente sintassi:

  ```
  $ grep [opzioni] pattern [files]
  ```

  dove pattern è un'espressione regolare
- Se il file non è specificato grep opera sullo standard input
- Di default sullo standard output vengono visualizzate le righe del testo che contengono almeno una stringa rappresentata dalla espressione regolare

Principali opzioni di grep

- -v stampa le righe che non contengono stringhe corrispondenti al pattern
- -n antepone il numero di riga
- -c visualizza soltanto quante righe contengono stringe corrispondenti al pattern
- -i rende il comando case-insensitive
- --color evidenzia, colorandola, l'occorrenza del pattern all'interno della riga

Espressioni Regolari

- Il caso più elementare di espressione regolare è un singolo carattere che rappresenta l'insieme costituito dal solo carattere
- Il carattere a ad esempio rappresenta l'insieme di stringhe costituito dal solo carattere {a}
- Il comando

```
$ grep a file
```

- Mostra tutte le linee del file che contengono la lettera a almeno una volta in qualsiasi punto della riga

Il carattere speciale punto

- Il carattere punto `.` è un carattere speciale che rappresenta qualsiasi carattere (lettera, cifra, spazio, tab, ...)
- Si noti che la sintassi delle espressioni regolari differisce da quella del file globbing dove un singolo carattere è rappresentato dal carattere '?'
- Il comando

  ```
  $ grep . file
  ```

  mostra tutte le linee del file che contengono almeno un carattere

Insiemi di caratteri

- Per identificare diversi caratteri è possibile utilizzare le parentesi quadre [ ] come accade per il file globbing
- Una lista di caratteri racchiusa tra parentesi quadre corrisponde a qualsiasi carattere in essa contenuta
- Il comando

  ```
  $ grep '[abcd]' file
  ```

  mostra tutte le linee del file che contengono almeno una tra le lettere a,b,c,d

Quoting dell'espressione regolare

- Dal momento che l'espansione della shell avviene prima dell'esecuzione del comando è sempre opportuno inserire espressioni regolari che contengono caratteri speciali per la shell tra apici singoli evitando che la shell li interpreti
- Questo è necessario anche quando i caratteri speciali hanno la stessa funzione
- Cosa succede a questo comando:

```
$ grep [abcd] file
```

se nella directory corrente ci sono due file nominati a e b e non si utilizzano gli apici?

Quoting dell'espressione regolare

- Dal momento che l'espansione della shell avviene prima dell'esecuzione del comando è sempre opportuno inserire espressioni regolari che contengono caratteri speciali per la shell tra apici singoli evitando che la shell li interpreti
- Questo è necessario anche quando i caratteri speciali hanno la stessa funzione
- Cosa succede a questo comando:

```
$ grep [abcd] file
```

se nella directory corrente ci sono due file nominati a e b e non si utilizzano gli apici?

```
$ grep a b file
```

Insiemi di caratteri

- Se la lista viene preceduta dal simbolo ^ l'espressione corrisponde a qualsiasi carattere non contenuto nella lista
- Il comando

```
$ grep '[^abcd]' file
```

  mostra le linee del file che contengono lettere diverse da a,b,c,d
- Esempi:

fughe	mostrata perché contiene f,u,g,h,e
coda	mostrata perché contiene o
bada	non mostrata perché non contiene lettere diverse da a,b,c,d

Intervalli di caratteri

- All'interno delle parentesi quadre è possibile indicare un intervallo di caratteri specificando gli estremi e separandoli con il carattere
- Il comando:

  $ grep '[a-g]' file

  mostra tutti 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]

Classi di caratteri

- Le parentesi quadre possono anche identificare una classe di caratteri
  [:alnum:] cifra o lettera
  [:alpha:] lettera
  [:digit:] cifra
  [:lower:] lettera minuscola
  [:print:] carattere stampabile
  [:upper:] lettera maiuscola
  [:xdigit:] cifra esadecimale
  ...
- Il comando:

```
$ grep [[:lower:]] file
```

  stampa tutte le righe che contengono una minuscola

Concatenazione

- Due espressioni regolari possono essere concatenate in una nuova espressione regolare
- Ad essa corrisponde qualsiasi stringa formata da due stringhe concatenate corrispondenti alle singole espressioni di partenza
- $ grep 'regexpr1regexpr2'

  stringa1stringa2

Concatenazione

- La più semplice forma di concatenazione è una stringa di lettere o numeri
- Il comando:

```
$ grep ab file
```

- stampa tutte le linee del file che contengono la stringa ab
- Il comando

```
$ grep 'a[1-5]' file
```

corrisponde a qualsiasi stringa {a1,a2,a3,a4,a5}

Posizione

- I caratteri ^ e $ identificano all'interno di un'espressione regolare rispettivamente inizio e fine riga
- Il comando:

```
$ grep 'sringa$' file
```

  stampa tutte le righe che terminano con stringa
- Il comando:

```
$ grep '^stringa' file
```

  stampa tutte le righe che iniziano con stringa
- Il comando:

```
$ grep '^stringa$' file
```

  stampa solo le righe che contengono soltanto stringa

Inizio e fine di una parola

- I simboli `\<` e `\>` individuano rispettivamente l'inizio e la fine di una parola
- Il comando:

```
$ grep '\<stringa'
```

  stampa la tutte le righe che contengono una parola che comincia con stringa (stringa, stringato, ma non costringa)

- Il comando:

```
$ grep 'stringa\>'
```

  stampa la tutte le righe che contengono una parola che termina con stringa (stringa, costringa, ma non stringato)

Ripetizioni (*)

- Il carattere * viene specificato per cercare un'espressione ripetuta un numero qualsiasi di volte (anche zero)
- Il comando:

```
$ grep 'ab*c'
```

- Visualizza le righe che contengono le stringe
  ac
  abc
  abbc
  abbbc
  ab...bc
- Attenzione non ha niente a che fare con il carattere * del file globbing!!!

Caratteri speciali all'interno di [...]

- I caratteri speciali all'interno di un'espressione regolare con le quadre perdono di significato e vengono considerati letteralmente:
- Il comando

```
$ grep   '[\.*]'
```

  mostra le linee del file che contengono i caratteri .,\ e * e non zero o più occorrenze del carattere .
- Esempio:

file.txt	mostrata perché contiene .
ciao a *	mostrata perché contiene *
exit	non mostrata perché non contiene \,. oppure *

Il carattere ^ all'interno di [...]

- Il carattere ^ se specificato dopo il primo carattere assume significato lettereale:
- Il comando

```
$ grep   '[a^b]'
```

  mostra le linee del file che contengono i caratteri a,^ e b e non caratteri a seguiti da caratteri diversi da b

- Esempio:

^_^	mostrata perché contiene ^
at	mostrata perché contiene a e ^
done	non mostrata perché non contiene a,^ oppure b

Espressioni regolari estese

- I caratteri ?, +, {, |, (, e) consentono di specificare ulteriori proprietà delle espressioni regolari
- Se utilizzati con grep (in assenza dell'opzione -E) che interpreta espressioni regolari base, questi caratteri vanno preceduti dal carattere '\'
- Se si utilizzano con egrep o con l'opzione -E di grep che abilita l'interpretazione di espressioni regolari estese non è necessario anteporre il carattere '\'
- La differenza è solo nella sintassi dell'espressione regolare non nella sua funzionalità:

  $ grep 'a\+' equivale a $ egrep 'a+'

Ripetizioni ?

- Il carattere ? viene specificato per cercare un'espressione ripetuta al più una volta (anche zero)
- Il comando:

```
$ egrep 'ab?c'
```

- Visualizza le righe che contengono le stringe
  ac
  abc
- Ma non corrisponde alle stringhe
  abbc
  abbbc
  ab...bc

Ripetizioni +

- Il carattere + viene specificato per cercare un'espressione ripetuta almeno una volta
- Il comando:

```
$ egrep 'ab+c'
```

- Visualizza le righe che contengono le stringhe
  abc
  abbc
  abbbc
  ab...bc
- Ma non corrisponde alla stringa
  ac

Ripetizioni {n}

- L'espressione {n} viene specificato per cercare un'espressione ripetuta esattamente n volte
- Il comando:

```
$ egrep 'ab{2}c'
```

- Visualizza le righe che contengono le stringhe
  abbc
  Ma non corrisponde alla stringa
  ac
  abc
  abbbc
  ab...bc

Ripetizioni {n,m}

- L'espressione {n,m} viene specificato per cercare un'espressione ripetuta almeno n volte e alpiù m volte
- Il comando:

```
$ egrep 'ab{2,5}c'
```

- Visualizza le righe che contengono le stringhe
  abbc
  abbbc
  abbbbc
  abbbbbc
  Ma non corrisponde alla stringa
  ac
  abc
  abbb...bbc

Ripetizioni {n,}

- L'espressione {n,} viene specificato per cercare un'espressione ripetuta almeno n volte
- Il comando:

```
$ grep '\<[[:digit:]]{4,}\>'
```

- Visualizza le righe contenenti i numeri con più di quattro cifre

Alternanza di Espressioni Regolari

- Due espressioni regolari possono essere concatenate utilizzando l'operatore pipe |
- L'espressione regolare risultante corrisponde a qualsiasi stringa corrisponda alla prima o alla seconda
- L'insieme delle stringhe rappresentate dall'espressione expr1|expr2 e l'unione degli insiemi delle stringhe rappresentata da expr1 e da expr2

Esempi di Alternanza

- Il comando:

  $ egrep 'a|b'

- Visualizza le righe che contengono le stringhe che contengono la lettera a e quelle che contengono la lettera b

Ordinamento nell'interpretazione

- 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

Esplicitando l'ordinamento

- Il comando
- $ egrep 'a|bc+'

  esplicitando l'ordinamento delle operazioni equivale a:

  $ egrep '(a|(b(c+)))'

  1) ripetizione

  2) concatenazione

  3) alternanza
- Cerchiamo stringhe del tipo
  a
  bcc....

Modificando l'ordinamento

- Modificando l'ordinamento delle operazioni equivale con le parentesi tonde:

  $ egrep 'a|(bc)+'

  1) concatenazione

  2) ripetizione

  3) alternanza

- Cerchiamo stringhe del tipo
  a
  bcbcbc....

Modificando l'ordinamento

- Modificando l'ordinamento delle operazioni equivale con le parentesi tonde:

  $ egrep '(a|b)c+'

  1) alternanza

  2) ripetizione

  3) concatenazione

  Cerchiamo stringhe del tipo
  acccccc...
  bccccc...

sed

- sed (stream editor) è uno strumento automatico per la modifica di file di testo
- La sintassi tipicamente utilizzata è

```
$ sed script [file]
```

- In assenza di un file come argomento viene processato lo standard input
- Di default il testo modificato viene visualizzato sullo standard output

Sostituzione di testo

- Il comando di sostituzione consente di rimpiazzare il testo corrispondente ad una espressione regolare con un altro testo
- La sintassi da utilizzare è:

```
$ sed 's/regexpr/rimpiazzo/'
```

- L'espressione regolare che si possono utilizzare di default sono quelle regolari (in cui i caratteri speciali ?, +, {, |, (, e) vanno preceduti dal carattere \
- L'opzione -r consente di utilizzare le espressioni regolari estese e di omettere il carattere \

Esempio di sostituzione del testo

- Il comando:

```
$ sed 's/^/Utente /' /etc/passwd
```

antepone ad ogni riga del file passwd la stringa “Utente “ trasformando ad esempio la riga:

```
oliva:x:1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

- Nella riga:

```
Utente oliva:x:1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

Delimitatori alternativi

- In alternativa al carattere / per delimitare i campi del comando di sostituzione si possono utilizzare i caratteri ':','_' oppure '|':
- Le sintassi:

  $ sed 's/1/uno/'
  $ sed 's_1_uno_'
  $ sed 's:1:uno:'
  $ sed 's|1|uno|'

  sono del tutto equivalenti.
- Comodo nel caso di sostituzione di path:

  ```
  $ sed 's_/usr/bin/_/usr/local/bin/_'
  ```

  che altrimenti potrebbe essere:

  ```
  $ sed 's/\/usr\/bin\///\/usr\/local\/bin\///'
  ```

Riutilizzare il testo trovato

- Con il carattere & è possibile riutilizzare il testo trovato mediante espressione regolare
- Il comando:

```
$ sed 's/[[:digit:]]\+:/con uid &/' /etc/passwd
```

trasforma la riga:

```
oliva:x:1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

- Nella riga:

```
oliva:x:con uid 1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

Riutilizzare parte del testo trovato

- Con l'espressione \n è possibile riutilizzare parte del trovato racchiuso tra parentesi tonde:
  \1 corrisponde alla prima parentesi
  \2 corrisponde alla seconda parentesi

  ...
- Il comando:

```
$ sed 's/:x:\([[:digit:]]\+\):/ con uid \1/' /etc/passwd
```

  trasforma la riga:

```
oliva:x:1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

- Nella riga:

```
oliva con uid 1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

Sostituzione globale

- sed di default sostituisce soltanto la prima occorrenza del testo nella riga
- Per sostituire tutte le occorrenze si usa il flag g posto dopo il carattere di delimitazione del rimpiazzo

```
$ sed 's/regexpr/rimpiazzo/g'
```

Sostituzione globale

- Il comando:

```
$ sed 's/,//' /etc/passwd
```

trasforma la riga:

- `oliva:x:1000:1000:Gennaro Oliva,,,:/home/oliva:/bin/bash`

in:

```
oliva:x:1000:1000:Gennaro Oliva,,:/home/oliva:/bin/bash
```

- Mentre il comando:

```
sed 's/,//g' /etc/passwd
```

la trasforma in in:

```
oliva:x:1000:1000:Gennaro Oliva:/home/oliva:/bin/bash
```

Operare su una precisa occorrenza

- Se una data espressione regolare da ricercare all'interno del testo occorre più di una volta all'interno di una linea è possibile specificare che la sostituzione avvenga sull' n-ima occorrenza
- Per sostituire l n-ima occorrenza si utilizza flag n (con n numero interno e non lettera) posto dopo il carattere di delimitazione del rimpiazzo

```
$ sed 's/regexpr/rimpiazzo/n'
```

Operare su una precisa occorrenza

- Il comando:

```
$ sed 's/[[:digit:]]\+/con gid &/2' /etc/passwd
```

  trasforma la riga:

```
oliva:x:1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

- Nella riga:

```
oliva:x:1000:con gid 1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

Effettuare più sostituzioni

- sed consente di effettuare più sostituzioni con la stessa esecuzione utilizzando l'opzione -e

```
$ sed -e 's/reg1/rim1/' -e 's/reg2/rim2/' \
   -e 's/reg3/rim3/' …
```

- Le sostituzioni vengono effettuate secondo l'ordine in cui sono state specificate sulla linea di comando: da sinistra verso destra
- La ricerca dell'espressione regolare di una sostituzione successiva alla prima viene effettuata sulle linee modificate dalle sostituzioni precedenti

Effettuare più sostituzioni

- Il comando:

```
$ sed -e 's/^/Utente /' \
-e 's/:x:\([[:digit:]]\+\)/ con uid \1/' \
-e 's/[[:digit:]]\+/con gid &/2' \
-e 's/:/ /' -e 's/:/ con GECOS /' \
-e 's/,//g' \
-e 's/:/ ha home directory in /' \
-e 's/:.*\/\(.*\)/ e usa la shell \1/' \
 /etc/passwd
```

- Modifica la riga:

```
oliva:x:1000:1000:Gennaro
Oliva,,,:/home/oliva:/bin/bash
```

  nella riga:

- `Utente oliva con uid 1000 con gid 1000 con GECOS Gennaro Oliva ha home directory in /home/oliva e usa la shell bash`

Cancellazione di testo

- Oltre a cancellare parti di testo sostituendole con una stringa vuota, con sed è possibile effettuare cancellazione di intere linee
- Il comando

  `$ sed '`*n*`d'`

  dove *n* è un intero, cancella l'*n*-ima riga, mentre
- `$ sed '/regexpr/d'`

  cancella tutte le righe che contengono l'espressione regolare regexpr