Ambienti di Programmazione per il Software di Base

Shell Script

Grep

grep [options] pattern [file]

Es. cat file.txt | grep ‘pattern’

grep ‘pattern’ file.txt

The grep utility searches any given input files, selecting lines that match one or more patterns. By default, a pattern matches an input line if the regular expression (RE) in the pattern matches the input line without its trailing newline. An empty expression matches every line. Each input line that matches at least one of the patterns is written to the standard output.

Sed

sed [options] command [file]

Es. cat file.txt | sed command

sed command file.txt

The sed utility reads the specified files, or the standard input if no files are specified, modifying the input as specified by a list of commands. The input is then written to the standard output.

Substitute: ‘s/pattern/replace/’

Delete: ‘/pattern/ d’

Regex

. (dot) - a single character.

? - the preceding character matches 0 or 1 times only.

***** - the preceding character matches 0 or more times.

+ - the preceding character matches 1 or more times.

{n} - the preceding character matches exactly n times.

{n,m} - the preceding character matches at least n times and not more than m times.

[agd] - the character is one of those included within the square brackets.

[^agd] - the character is not one of those included within the square brackets.

[c-f] - the dash within the square brackets operates as a range. In this case it means either the letters c, d, e or f.

() - allows us to group several characters to behave as one.

| (pipe symbol) - the logical OR operation.

^ - matches the beginning of the line.

$ - matches the end of the line.

Shell: Espressioni Regolari

Esempio: grep ^*public* *.java (trova tutte le righe che iniziano con public)

Operator	Effect
.	Matches any single character.
?	The preceding item is optional and will be matched, at most, once.
*	The preceding item will be matched zero or more times.
+	The preceding item will be matched one or more times.
{N}	The preceding item is matched exactly N times.
{N,}	The preceding item is matched N or more times.
{N,M}	The preceding item is matched at least N times, but not more than M times.
-	represents the range if it's not first or last in a list or the ending point of a range in a list.
^	Matches the empty string at the beginning of a line; also represents the characters not in the range of a list.
$	Matches the empty string at the end of a line.
\b	Matches the empty string at the edge of a word.
\B	Matches the empty string provided it's not at the edge of a word.
\<	Match the empty string at the beginning of word.
\>	Match the empty string at the end of word.

Shell: sed (1)

Comandi di Editing di Sed

Command	Result
a\	Append text below current line.
c\	Change text in the current line with new text.
d	Delete text.
i\	Insert text above current line.
p	Print text.
r	Read a file.
s	Search and replace text.
w	Write to a file.

Option	Effect
`-e SCRIPT`	Add the commands in SCRIPT to the set of commands to be run while processing the input.
`-f`	Add the commands contained in the file SCRIPT-FILE to the set of commands to be run while processing the input.
`-n`	Silent mode.
`-V`	Print version information and exit.

Opzioni di sed

Shell: sed (2)

```
sandy ~> cat -n example
     1  This is the first line of an example text.
     2  It is a text with erors.
     3  Lots of erors.
     4  So much erors, all these erors are making me sick.
     5  This is a line not containing any errors.
     6  This is the last line.

sandy ~>
```

1)sed -n *'/^This.*errors.$/p'* example
This is a line not containing any errors.

2)sed *'3,$d'* example
This is the first line of an example text.
It is a text with erors.
3)**sed *'s/erors/errors/'* example**
This is the first line of an example text.
It is a text with errors.
Lots of errors.
So much errors, all these errors are making me sick.
This is a line not containing any errors.
This is the last line.

Shell: sed (3)

4) sed *'s/erors/errors/g'* example

This is the first line of an example text.
It is a text with errors.
Lots of errors. So much errors, all these errors are making me sick.
This is a line not containing any errors.
This is the last line.

5) sed *'s/^/> /'* example

This is the first line of an example text.
> It is a text with erors.
> Lots of erors.
> So much erors, all these erors are making me sick.
>This is a line not containing any errors.
>This is the last line.

6) sed -e *'s/erors/errors/g'* -e *'s/last/final/g'* example

This is the first line of an example text.
It is a text with errors. Lots of errors.
So much errors, all these errors are making me sick.
This is a line not containing any errors.
This is the final line.

Grep

grep REGEX file.txt

1.cat file.txt | grep ‘^[0-9]’

2.cat file.txt | grep ‘txt$’

3.cat file.txt | grep ‘copia[0-9]+\.sh’

4.cat file.txt | grep ‘copia[^0-9]\.sh’

5.cat file.txt | grep ‘copia.*\.txt’

Sed

1. cat test.txt | sed 's/g/G/’
2. echo "nome.mp3" | sed 's/nome/cognome/’
3. echo "this is a file. this is a text" | sed 's/this/THIS/g’
4. cat test.txt | sed 's/\([a-c]\)/\U\1/’
5. cat test.txt | sed 's/[a-c]/999999/’
6. echo 'aaaabbbbcccc' | sed 's/a*/99/’
7. seq -s ' ' 1 9 | sed 's/[3-5]/A/g’
8. seq -s ' ' 1 9 | sed 's/[^3-5]/A/g’

Esercizio

Scrivere un programma shell (**copia_modificati.sh**) che riceva come parametri un file di testo e due cartelle. Esempio d'uso sarà: **copia_modificati.sh file_da_copiare.txt cartellaA cartellaB**.

Il file conterrà per ogni riga il nome di un file (che è presente nella cartella **cartellaA**).
Per ogni file letto dal file **file_da_copiare.txt**, tale file dovrà essere copiato dalla cartella **cartellaA** alla cartella **cartellaB** (attenzione, i nomi delle cartelle sono passati come parametri, quindi non devono per forza essere cartellaA e cartellaB!!!!!). Alla fine si dovrà stampare in output il numero e il nome dei file copiati nella **cartellaB**.

Gestire anche il controllo degli errori (parametri insufficienti, cartella inesistente, ecc..).

Esercizio

Scrivere un programma shell (**copia_modificati.sh**) che riceva come parametri un file di testo e due cartelle. Esempio d'uso sarà: **copia_modificati.sh file_da_copiare.txt cartellaA cartellaB**.

Il file conterrà per ogni riga il nome di un file (che è presente nella cartella **cartellaA**).
Per ogni file letto dal file **file_da_copiare.txt**, tale file dovrà essere copiato dalla cartella **cartellaA** alla cartella **cartellaB** (attenzione, i nomi delle cartelle sono passati come parametri, quindi non devono per forza essere cartellaA e cartellaB!!!!!). Se il file è già presente nella cartella **cartellaB**, prima di copiarlo bisognerà chiedere all'utente "Vuoi sovrascriverlo?" e in caso di risposta affermativa bisognerà copiarlo e creare una versione di backup della copia pre-esistente (sostituendo l'estensione originale con .bak).

Alla fine si dovrà stampare in output il numero e il nome dei file copiati nella **cartellaB** e di quelli che non sono stati copiati. Gestire anche il controllo degli errori (parametri insufficienti, cartella inesistente, ecc..).

Esercizio

Scrivere un programma shell che esegua le operazioni:

1.Legge tutti in file .txt nella directory specificata come primo argomento

2.Per ogni file trovato, converte il primo carattere di ogni riga in maiuscolo e salva il risultato in un file con lo stesso nome memorizzato nella directory specificata come secondo argomento.

3.Stampa il numero di file letti/modificati.

Es.

capitalize.sh cartellaA cartella B

Esercizio

Scrivere un programma shell che, date in input due cartelle A e B, raggruppa i file contenuti in A per tipologia. Per ogni file contenuto in A, crea una directory con nome pari all'estensione del file (da creare nella cartella B) ed effettua la copia del file.

Es.

A

file1.txt

file2.txt

file3.mp3

file4.mkv

B

txt

file1.txt

file2.txt

mp3

file3.mp3

mkv

file4.mkv