

La Shell di UNIX

2 – Comandi di UNIX

Dipartimento di Informatica
Università di Pisa
Largo B. Pontecorvo 3
56127 Pisa



Temi

- Ambiente di Lavoro (2h)
- Comandi di UNIX (2h)
- Script di Shell (4h)
- (Emacs Beginner's HOWTO)

Overview

- 1 Navigare nel filesystem
 - Alcuni comandi base
 - Collegamenti
 - Permessi dei file
 - Altri comandi utili
- 2 Stream EDitor (SED)

- 1 Navigare nel filesystem
 - Alcuni comandi base
 - Collegamenti
 - Permessi dei file
 - Altri comandi utili

- 2 Stream EDitor (SED)

Login - di nuovo

<https://bellard.org/jslinux/> - Alpine Linux

```
Loading...
Welcome to JS/Linux (i586)

Use 'vlogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export_file filename' to export a file to your computer.
Imported files are written to the home directory.

localhost:~# vlogin HandsOn
Password:
Your account is now synchronized with the server every 60 seconds. You can force
a synchronization with the 'vfsync' shell command.

localhost:~$ █
```

Visualizzare il contenuto di una directory (1)

```
ls [-alsFR] [<dir1> ... <dirN>]
```

Se non viene specificata alcuna directory, si riferisce alla directory corrente.

Alcune opzioni:

- a** visualizza anche i file nascosti (il loro nome inizia per “.”);
- l** visualizza informazioni estese sui file (es. permessi, dimensione, owner, group);
- s** visualizza la dimensione in bytes;
- F** aggiunge un carattere finale al nome del file che ne denota il tipo (es. “nome/” indica una directory);
- R** visualizza ricorsivamente le sottodirectory (esegue `ls` ricorsivamente sulle subdir).

Visualizzare il contenuto di una directory (2)

Esempio

```
$ ls -al
```

```
drwxrwxr-x 3 HandsOn user2 4096 Jun 28 20:24  
Data
```

- tipo di file: `d` (directory), `-` file, `l` link simbolico
- i permessi (3x3 proprietario/gruppo/gli altri),
- numero di hard link (3),
- nome del proprietario (HandsOn),
- gruppo del proprietario (user2),
- file/directory dim. in bytes (4096),
- momento di ultima modifica (Jun 28 20:24).

I metacaratteri (wildcards)

I seguenti metacaratteri, chiamati **wildcard** sono usati per **abbreviare** i nomi di file in un pathname:

'?' qualunque carattere in quella posizione del nome del file

'*' qualunque **sequenza** di caratteri

'*<exp>*' corrisponde a **tutti** i caratteri iniziali e finali del file con `exp` in mezzo

'<char>*' corrisponde all **range** di caratteri tra parantesi.



Esempi: `ls ?a?` elenca tutti i nomi di file di lettere con 'a' nel mezzo

`ls *abc*` ... con 'abc' nel mezzo

`ls [a-e]*` ... con i caratteri iniziali a,b,c,d,e

`ls *. [xyz]*` ... con estensione che inizia con x, y o z.

Nomi di file UNIX

- I nomi di file UNIX possono contenere qualsiasi carattere della tastiera eccetto '/' fino a 256 caratteri, inclusi i caratteri speciali '*', '?', '#', '&' e spazio vuoto (ma sono **difficili da usare**).
- **non può** contenere altri file o directory
- Scelta ottima: caratteri alfanumerici combinato da _ (trattino basso) e . (periodo).

Creare una directory

```
mkdir [-p] <dir1> ... <dirN>
```

I parametri **dir** indicano i nomi (path assoluti o relativi) delle directory da creare.

Opzioni:

-p crea eventuali directory intermedie esplicitate nei parametri **dir**.

Esempio

- **mkdir temp** — crea directory **temp** nella directory corrente.
- **mkdir -p docs/personal** — crea le directory **personal** dentro la dir **docs** (se **docs** non esiste viene creata).

Cambiare una directory

`cd [<dir>]` serve per muoversi attraverso le directory.

Il parametro `<dir>` è opzionale — se non viene indicato, il comando porta nella home directory.

Esempio

- Supponiamo che vogliamo accedere ai nostri documenti personali in `/home/user/docs`
- supponiamo che la directory corrente sia la nostra home: `/home/user`
- per portarsi nella directory dei documenti eseguire:
`cd docs`

Eliminare una directory

```
rmdir [-p] <dir1> ... <dirN>
```

- I parametri **dir** indicano i nomi (pathname assoluti o relativi) delle directory da eliminare.
- Opzioni
 - **-p** elimina eventuali directory intermedie esplicitate nei pathname dei parametri **dir**.

Esempio

- **rmdir temp** — elimina la directory **temp** **se è vuota**.
- **rmdir -p docs/personal** — elimina le directory **personal** e **docs**, se entrambe vuote.

Creare un file vuoto

```
touch <file1> <file2> ...
```

- I parametri file indicano i nomi dei file o directory da aggiornare.
- Se un file non esiste, il comportamento predefinito è quello di creare un nuovo file vuoto.

Esempio

```
$ touch prova.txt
```

Copiare file

```
cp [-if] <file1> <file2>
```

copia **file1** in **file2** — se **file2** esiste, viene sovrascritto!

```
cp [-if] <file1> ... <fileN> <dir>
```

copia i **file** nella directory **dir** — se un **file** esiste in **dir**, viene sovrascritto!

Opzioni:

- i** chiede conferma prima di sovrascrivere;
- f** non chiede conferma prima di sovrascrivere.

Spostare file

```
mv [-if] <file1> <file2>
```

sposta **file1** in **file2** — **se file2 esiste viene sovrascritto!**

```
mv [-if] <file1> ... <fileN> <dir>
```

sposta **file** nella directory **dir** — **se un file esiste in dir anche questa viene sovrascritto!**

Ozioni:

- i** chiede conferma prima di sovrascrivere;
- f** non chiede conferma prima di sovrascrivere.

Eliminazione di file

```
rm [-rif] <file1> ... <fileN>
```

Opzioni:

- r <dir>** cancella in modo ricorsivo la directory con il suo contenuto;
- i** prima di cancella il file chiede conferma all'utente;
- f** cancella senza chiedere conferma.

Echo (1)

- Il comando **echo** **<arg>** stampa sullo schermo la stringa passata come parametro **<arg>** al comando.

Esempi

```
$ echo Ciao!  
Ciao!
```

- Il comando **echo** **\$<arg>** stampa sullo schermo il contenuto del parametro **<arg>**.

Esempio

```
$ x=10  
$ echo $x  
10
```

Echo (2)

N.B.

In combinazione con i metacaratteri (jolly), `echo` elenca file e directory!

Esempi

```
$ ls  
data-new data1 data2 inittab example1.txt  
  
$ echo data*  
data-new data1 data2  
  
$ echo data?  
data1 data2
```

Redirezione (1)

Di default i comandi Unix prendono l'input da tastiera (**standard input - stdin**) e mandano l'output ed eventuali messaggi di errore su video (**standard output - stdout**, **standard error - stderr**). L'input/output in Unix può essere rediretto da/verso file, utilizzando opportuni metacaratteri:

Metacarattere	Significato
>	ridirezione dell'output (new)
>>	ridirezione dell'output (append)
<	ridirezione dell'input (new)
<<	ridirezione dell'input dalla linea di comando (append)

Redirezione (2)

Esempi

```
$ echo shell programming in UNIX > file.txt
```

```
$ cat file.txt
```

```
shell programming in UNIX
```

```
$ echo makes fun >> file.txt
```

```
$ echo $(<file.txt)
```

```
shell programming in UNIX makes fun
```

```
$ cat list1 list2 > biglist
```

Backtick

Abbiamo visto i metacaratteri '?', '*' and '\$'. C'è un altro tipo in UNIX: il **backtick** (`).

Il testo tra i backtick viene eseguito e sostituito dall'output del comando

Esempio

```
$ hostname
```

```
Desktop-INF
```

```
$ echo my machine is called `hostname`
```

```
my machine is called Desktop-INF
```

N.B.

Il codice ALT+96 corrisponde a backtick sulla tastiera italiano standard.

Pipe

- Il metacarattere “|” (**pipe**) prende l’output del comando alla sinistra del | e lo manda all’input del comando alla destra .
- Una **pipeline** è una sequenza di una o più pipe.
 - serve per comporre comandi “in cascata” in modo che l’output di ciascuno sia fornito in input al successivo.
 - L’output dell’ultimo comando è l’output della pipeline (di default sullo standard output).
 - La sintassi è:
`comando1 | comando2 | ... | comandoN .`

Esempio

```
ls | more
```

L’effetto è quello di visualizzare l’output di `ls` una pagina per volta.

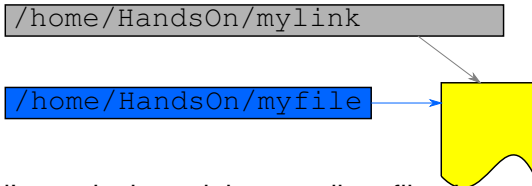
1 Navigare nel filesystem

- Alcuni comandi base
- Collegamenti
- Permessi dei file
- Altri comandi utili

2 Stream EDitor (SED)

Collegamenti (1)

Collegamento fisico (Hardlink)



- indica l'associazione del nome di un file al suo contenuto;
- deve essere creato all'interno dello stesso filesystem;
- l'hard link ha lo **stesso numero inode** del file originario;
- essendo una copia speculare di un altro file, **occupa spazio** nel filesystem;
- se si cancella, rinomina, sposta, il file originale, l'hard link **continuerà ad esistere**.

Sintassi

```
$ ln <source> <link>
```


Collegamenti (2)

Esempio

```
$ ln myfile mylink  
$ ls -il my*  
68719 -rw-r--r-- 2 HandsOn user2 17 set 25 09:57 myfile  
68719 -rw-r--r-- 2 HandsOn user2 17 set 25 09:57 mylink
```

Dallo stato del collegamento, è chiaro che

```
$ stat mylink  
File: 'mylink'  
Size: 17 Blocks: 8 IO Block: 4096 regular file
```

il collegamento è in fatti un file regolare.

Collegamenti (3)

Collegamento fisico per le directory ?

Considera il seguente scenario:

```
$ mkdir -p /tmp/a/b  
$ cd /tmp/a/b  
$ ln -d /tmp/a c  
ln: failed to create hard link 'c' =>  
'/tmp/a': Operation not permitted
```

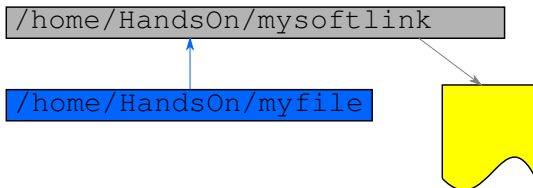
Due diverse directory in punti diversi nel filesystem **potrebbero puntare alla stessa cosa, creando un ciclo.**

```
$ cd /tmp/a/b/c/b/c/b/c/b/c/b
```

Questo tipo di filesystem con cicli non è più un albero!

Collegamenti (4)

Collegamento simbolico (Softlink)



- Un link simbolico è un file, contenente all'interno un collegamento ad un altro file (o directory);
- Il link simbolico e il file originale hanno **un numero inode differente**: quindi per il filesystem sono elementi diversi anche se legati da questo link;
- Un link simbolico quindi non contiene alcun dato, ma solo un collegamento;
- Possiamo creare **più link simbolici che puntano allo stesso** file originale.

Collegamenti (5)

Creiamo un link simbolico

```
$ ln -s <source> <link>
```

Per vedere dove punta il collegamento, digitiamo

```
$ stat <link>  
File:  '<link>' -> '<source>'  
Size:  5 Blocks:  0 IO Block:  4096 symbolic link
```

Collegamenti (6)

```
$ ln -s myfile mysoftlink
$ ls -il mysoftlink
68719 -rw-r--r-- 2 HandsOn user2 17 set 25 09:57 myfile
68743 lrwxrwxrwx 1 HandsOn user2 5 set 25 16.39 mysoftlink ->
myfile
$
```

- viene rappresentato come 1 nel primo permesso,
- numero di collegamenti non si cambia,
- i permessi di un collegamento simbolico sono pressochè irrilevanti (sembra `rwxrwxrwx`),
- i permessi del `myfile` determinano l'accesso,
- se il file a cui un collegamento simbolico punta **viene rimosso o cambiato di nome**, il collegamento rimane **orfano**.

Printing Working Directory (pwd)

visualizza il percorso assoluto della directory corrente;

pwd [-LP]

Opzioni:

- L Visualizza il **percorso logico** della dir di lavoro corrente, **con** il nome del collegamento simbolico [default];
- P Visualizza il **percorso fisico** della dir di lavoro corrente, **senza** il nome del collegamento simbolico.

Esempio

```
$ ln -s mydir mysoftlink
$ cd mysoftlink
$ pwd -L
/home/MyName/mysoftlink
$ pwd -P
/home/MyName/mydir
```

- 1 Navigare nel filesystem
 - Alcuni comandi base
 - Collegamenti
 - **Permessi dei file**
 - Altri comandi utili
- 2 Stream EDitor (SED)

Permessi dei file (1)

- Per fare modifiche agli attributi di file il comando da utilizzare nel terminale per la modifica dei permessi è
- solo il proprietario di questo file e root possono farlo.
- Sintassi:

chmod <options> <permissions> <file>

Opzioni:

- v** (`--verbose`) Visualizza informazioni dettagliate su ogni file che `chown` tenta di modificare;
- R** (`--recursive`) Applica le modifiche ai file e alle directory indicata e a tutte le sue sottodirectory;
- preserve-root** Non operare in modo ricorsivo su '/'.

Permessi dei file (2)

Permissions:

- vengono definiti con un numero ottale composta da tre cifre **o**
- vengono definiti nella forma: chi-opcode-permesso.
 - **u** (user), **g** (group), **o** (other), **a** (all)
 - **r** (read) [4], **w** (write) [2], **x** (execute) [1]
 - **+** (add), **-** (remove), **=** set.

Esempio

```
$ chmod ug=rw,o-rw,a-x *.txt
```

imposta i permessi su tutti i file che terminano con `*.txt` su
`rw-rw----`

E questo ?

```
$ chmod 660 *.txt
```

Il gruppo associato a un file

- I gruppi sono liste di utenti che condividono le stesse autorizzazioni e permessi di `rxw`.
- Quelli utenti **condividono privilegi, sicurezza e accesso** in un sistema **multiutente** (come UNIX)
- Come cambiare gruppo a file e directory ?
`chgrp -R <new_group> <file/directory>`
- può anche essere modificato in modo ricorsivo con l'opzione `-R`.

Esempio

```
$ chgrp mystudents homework
```

Tutti gli studenti del gruppo `mystudents` possono accedere la directory `homework` allo **stesso** momento.

- 1 Navigare nel filesystem
 - Alcuni comandi base
 - Collegamenti
 - Permessi dei file
 - Altri comandi utili

- 2 Stream EDitor (SED)

Visualizzare il contenuto di un file (1)

```
cat [-nve] <file1> ... <fileN>
```

- Opzioni
 - n** precede ogni linea con un numero;
 - v** visualizza i caratteri non stampabili eccetto newline, tab e form-feed (un carattere speciale che salta una pagina);
 - e** visualizza \$alla fine di ogni linea (quando usato insieme con l'opzione -**v**);
- **cat file1 file2 file3** concatena il contenuto dei file seguendo lo stesso ordine di immissione e ne mostrerà il contenuto;

N.B.

Il comando **cat** visualizza il contenuto di un **intero file** (potenzialmente riempiendo il buffer dello schermo).

Visualizzare il contenuto di un file (2)

- **head -n**

visualizza le prime **n** linee di un file di testo

es., `head -10 example.txt` visualizza le prime 10 linee di `example.txt`;

- **tail**

-n visualizza le ultime **n** linee

es., `tail -10 example.txt` — visualizza le ultime 10 linee di `example.txt`;

-f permette di seguire in tempo reale le aggiunte in coda ad un file es, `/var/log/auth.log`.

Visualizzare il tipo di un file

- **file** è un comando, che cerca di determinare il **tipo** di uno o più file a partire **dal loro contenuto**.
- Sintassi:

file <file>

fornisce informazioni sulla natura dello stesso.

Esempio

```
$ file lecture1.pdf
```

```
lecture1.pdf: PDF document, version 1.4
```

```
$ file background.jpg
```

```
background.jpeg: JPEG image data, JFIF  
standard 1.01, aspect ratio, density 1x1,  
segment length 16, baseline, precision 8,  
2048x1371, frames 3
```

Trova File e Applicazioni (1)

- Dato (approssimativamente) il nome di un file, la sua posizione può essere estratta da

```
find -RLP <directory> -name <file>
```

a partire dalla directory reale, `directory`.

- Opzioni

- R permette all'utente di scendere ricorsivamente nelle sottodirectory;
- L In caso di collegamenti simbolici, sono presi in considerazione gli attributi dei file e/o delle directory;
- P In caso di collegamenti simbolici, **non** sono presi in considerazione gli attributi [default].

Esempio

```
$ find /usr/bin -name Foxit*  
/usr/bin/FoxitReader  
/usr/bin/FoxitReader.sh
```

Trova File e Applicazioni (2)

- Agisce su **un database** e quindi è **più veloce**;
- **non controlla se i file trovati nel db** esistono ancora e non segnala mai i file creati dopo l'aggiornamento più recente del relativo db;
- per aggiornare il dB, digi **updatedb**.
- Sintassi:

```
locate <filename>
```

Esempio

```
$ locate *FoxitReader*
```

Si produrrà un elenco dei luoghi dove si può trovare file denominati.

Ricerca di stringhe di caratteri

- Per ricercare una determinata **stringa di caratteri in un file** si utilizza la
- Sintassi:

```
grep [-n] <patterns> <file>
```

dove

-n il numero del righe,

patterns è la parola o la frase da ricercare e

file è il nome del file nel quale deve essere eseguita la ricerca.

- Uso di **grep** come filtro.

Esempio

```
$ ls -al *.txt | grep Apr
```

elenca tutti i file con estensione `txt` creati nel mese di aprile.

Ordinare un elenco

- **sort** legge uno o più file di testo/STDIN e ne **ordina le linee**, producendo il risultato su di un file/STDOUT;
- Sintassi

sort <option> <file1 file2 ...>

- k (--key) <value>: indica il campo da usare come chiave di l'ordimento;
- n (--numeric-sort): ordina numericamente invece che alfabeticamente.
- M (--month-sort): ordina per mese.
- u (--unique): elimina dal risultato le linee duplicate

Esempi

```
$ ls -al | sort -k5n -k9
```

Compressione e archiviazione dei file

- **gzip/gunzip** — GNU compressione/decompressione di file
es, `gzip example.txt` — ottengo il file compresso `example.txt.gz`;
- **bzip2/bunzip2** — Burrows-Wheeler compressione/decompressione di file;
- **tar** — creazione/estrazione da archivi;
- **zip/unzip** and **rar/unrar** — creazione e estrazione di archivi compressi.

- 1 Navigare nel filesystem
 - Alcuni comandi base
 - Collegamenti
 - Permessi dei file
 - Altri comandi utili
- 2 Stream EDitor (SED)

Stream EDitor **sed**

- da Lee E. McMahon, 1974 nei Bell Labs
- editor di linea che non richiede l'interazione con l'utente
- legge l'input, in modo predefinito, dallo `STDIN`
- può **filtrare** l'input che riceve da un file o una pipe
- fa uso delle **Espressioni Regolari**
- invia i risultati allo `STDOUT`
- `sed` **non** modifica l'input in modo predefinito
- Sito ufficiale:

<https://www.gnu.org/software/sed/>

sed

Consente di effettuare in modo non interattivo le seguenti operazioni:

- sostituire stringhe
- cancellare linee
- aggiungere linee
- rimpiazzare linee

Esempio tipico

"Il formato tempo non è corretto, manca una virgola tra i secondi e i millesimi di secondo! Vogliamo aggiungerla automaticamente"

Sintassi `sed`

- Sintassi

`sed [OPZIONI] [SCRIPT] [file di input]`

- Le `OPZIONI` riguardano la modalità di filtraggio e le operazioni (come ad esempio sostituzione del testo) da effettuare;
 - e (`--expression`) permette di specificare delle istruzioni e che si aggiungono alle altre eventualmente già indicate;
 - f (`--file`) permette di indicare un file contenente una serie di istruzioni `sed`;
 - i (`--insert`) aggiorna il file;
 - n (`--quiet`) stampa sullo `STDOUT` solo quando esplicitamente indicato tramite il comando `p`.
- Uno `SCRIPT` che deve essere interpretato direttamente da `sed`;
- Il `file di input`, opzionale, contiene il testo su cui saranno applicati i filtri.

Operatori `sed` di base

- a** (append) aggiunge di testo al di sotto della riga corrente;
- c** (change) Modifica il testo della riga corrente;
- d** (delete) Cancella testo;
- g** (global) Agisce su tutte le verifiche d'occorrenza di ogni riga di input controllata;
- i** (insert) Inserisci testo al di sopra della riga corrente;
- p** (print) Visualizza;
- s** (substitute) Sostituisce in ogni riga un'espressione regolare con un'altra;
- r** (read) legge un file e aggiungerlo al fine della riga.

Espressioni regolari in `sed`

Operatore	Descrizione
\	<i>Escape</i> protegge il carattere seguente da un'interpretazione diversa da quella letterale,
^	L'inizio della riga,
.	corrisponde a un carattere qualunque,
\$	fine riga,
()	definiscono un raggruppamento.
[xy]	un elenco di caratteri alternativi.
[x-y]	un intervallo di caratteri alternativi.
[^...]	I caratteri che non appartengono all'insieme.

sed DEMO

Esempi

```
myfile.txt
```

```
$ nano myfile.txt
```

```
"Le fragole sono buone"
```

```
Le fragole sono belle,
```

```
Amo le fragole.
```

```
Ctrl O, Ctrl X
```

```
$ cat myfile.txt
```

Cancellazione della righe

```
$ sed -e '5d' myfile.txt
```

```
$ sed -e '1d;5d' myfile.txt
```

sed DEMO

Esempi

Aggiungere riga (1)

```
$ sed '/belle/a sono di Pisa' myfile.txt
```

Aggiungere riga (2)

```
$ search='belle'  
$ add='sono di Pisa'  
$ sed -e "/${search}/a${add}" myfile.txt
```

aggiorna il file

```
$ sed -i "${search}/a${add}"
```

Visualizza solo le righe in cui è presente "Amo" (con l'opz. -n)

```
$ sed -n '/Amo/p' myfile.txt
```

sed DEMO

Esempi

Selezionare le righe dispari

```
$ sed -n -e '<FIRST>~<STEP>p' myfile.txt  
$ sed -n -e '1~2p' myfile.txt
```

Selezionare le righe pari

```
$ sed -n -e '0~2p' myfile.txt
```

Selezionare un intervallo

```
$ sed -e '5,10d' myfile.txt - sono cancellate le righe  
numerate da 5 a 10 comprese.  
$ sed -n -e '1,/^\$/p' myfile.txt - sono stampato  
tutte le righe a partire dalla n. 1 fino alla prima riga vuota.
```

sed DEMO

Esempi

Sintassi generale dell'istruzione di sostituzione

' s/pattern/replacement/flag'

- **pattern** serve per individuare quale sia la stringa da sostituire
- **replacement** rappresenta la sequenza di caratteri che sostituisce la **pattern**
- **flag** definiscono particolari modalità di ricerca o di sostituzione delle stringhe

Selezionare e sostituire *una* intera riga non vuota / *due* 'll' nella stringa

```
$ sed -e 's/.\{\1,\}/replacement/' myfile.txt  
$ sed -e 's/l\{\2,\}/replacement/' myfile.txt
```

sed DEMO

Esempi

Una semplice sostituzione

```
$ sed -e 's/ fragole / arance /' myfile.txt
```

Una sottostringa composta dalla sequenza di caratteri 'fragole' viene sostituita con la stringa 'arance'

Aggiungere una stringa dopo la stringa selezionata con il pattern

```
$ sed -e 's/[Ll]e / &mie /' myfile.txt
```

sed cerca le sottostringhe 'Le' o 'le'. Dopo aver trovato aggiunge 'mie' a loro grazie al metacarattere &.

sed DEMO

Esempi

- Il metacarattere di escape \

Togliere il significato speciale agli altri metacaratteri

```
$ sed -e 's /fragole\\/fragole\\/' myfile.txt
```

- il metacarattere \N (un valore numerico)

Selezionare e ripetere una porzione di stringa

```
$ sed -e 's/"\(.*\)" /@1@/' myfile.txt
```

seleziona un **gruppo** di caratteri qualsiasi delimitata all'inizio e alla fine da virgolette e sostituitala da una sequenza composta dal carattere @ (\1 punta al questo (primo) gruppo)

sed DEMO

Esempi

- il metacarattere `\L` – lowercase (minoscologia), `\U` – uppercase (maiuscola) e `\E` – end (fatto)

Esempio [non implementato nell'emulatore]

```
$ sed -e 's/\(.*\)/\L\1/' myfile.txt -
```

Conversione tutto in minuscola

```
$ sed -e 's/\(.*\)/\U\1/' myfile.txt -
```

Conversione tutto in maiuscola

```
$ sed -e 's/\(Le\)\(.*\)\(sono\)/\1\U\2\E\3/'  
myfile.txt - Conversione del 2o gruppo in maiuscola
```

- asterisco: zero o più volte.

sed DEMO

Esempi

- Aggiungere una colonna di numeri

Esempio

```
$ sed '=' myfile.txt | sed 'N;s/\n/ /'
```

- Questo pipe i dati dal primo comando in un altro comando **sed** che
- entra in modalità **multilinea (N)** e
- sostituire (s) tutti i caratteri di nuova riga (**\n**) con le schede (" "). Questo ha l'effetto di mettere i numeri di riga e il testo sulla stessa linea.

- 1 Navigare nel filesystem
 - Alcuni comandi base
 - Collegamenti
 - Permessi dei file
 - Altri comandi utili

- 2 Stream EDitor (SED)