

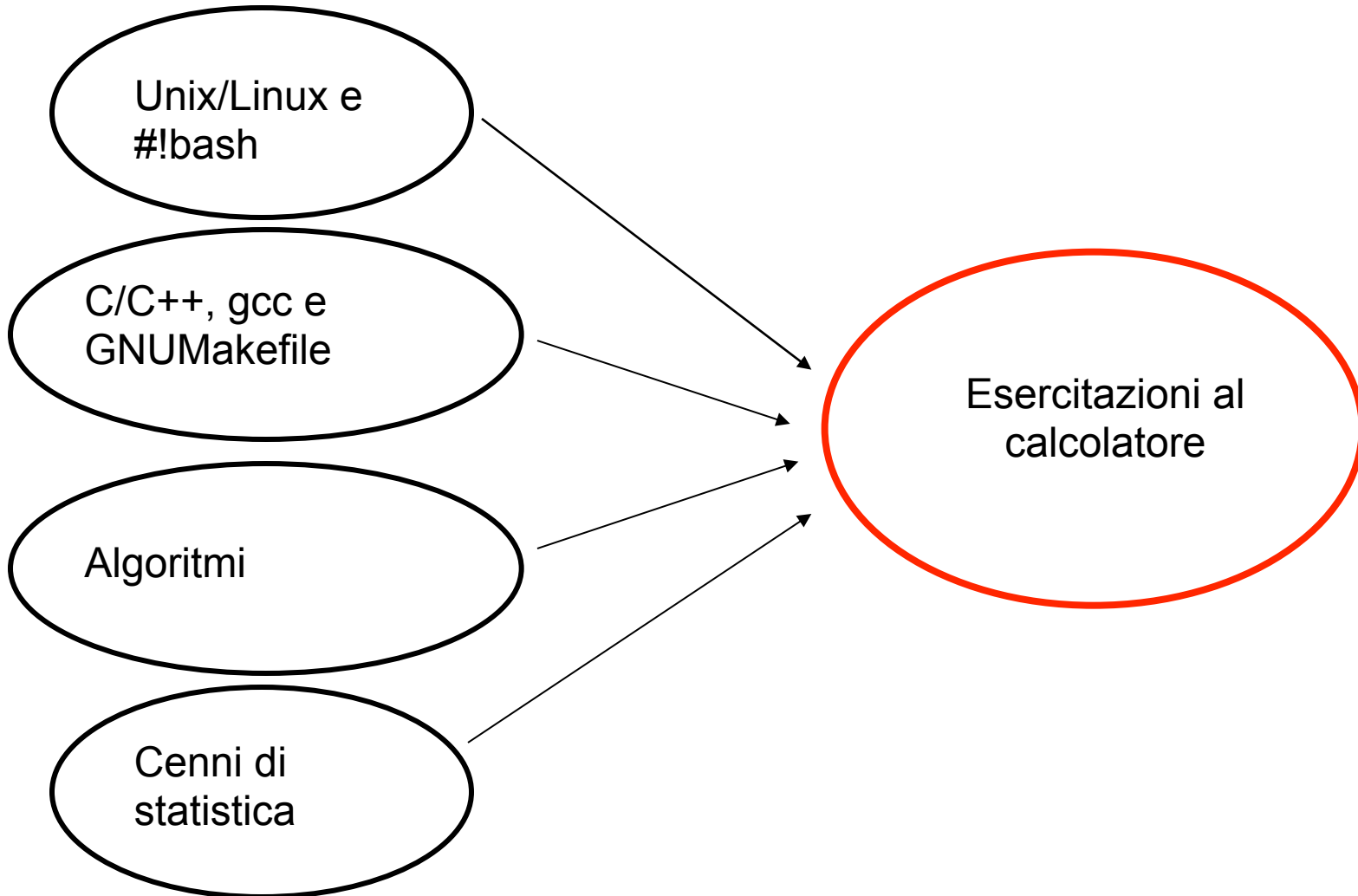
# Metodi Computazionali per la Fisica 2018-2019

Matteo Duranti

[matteo.duranti@pg.infn.it](mailto:matteo.duranti@pg.infn.it)

(cfr. <http://www.fisica.unipg.it/~borromeo/Appunti/FisComp/FisComp2014/pdf/linux.pdf>)

# Schema del corso



# Schema del corso

## “Informatica pura”:

- ~~utilizzo del sistema Unix/Linux e utilizzo di bash~~
- programmazione in C/C++, compilazione con gcc/g++ e uso di Makefile
- ricorsione, etc...

## “Algoritmi”:

- Integrazione numerica (Regola Trapezio, Simpson, Gauss)
- Risoluzioni sistemi di equazioni
- Equazioni differenziali (Eulero e Runge-Kutta)
- Interpolazione
- Minimizzazione (forse...)
- ...

## “Statistica”:

- Numeri random
- MonteCarlo e MCMC (forse)
- ...

# Lezioni

- Corso:
  - “Teoria”: ~ 10h
  - Esercitazioni: ~ 32h
  - Frequenza obbligatoria
- Ricevimento:
  - Giovedì: 11:00 – 13:00 (ma scrivetemi prima per mail!)
  - 5° piano del Dipartimento di Fisica
  - **leggete le mail (@studenti.unipg.it)!!!!**

# Lezioni

- Lezione:
  - Giovedì
    - 15.00 – 16.00, lezione frontale, Laboratorio di Informatica
  - Venerdì
    - 14.00 – 17.00, esercitazione al calcolatore, Laboratorio di Informatica

# Esame

- **Esame:**
  - relazione scritta su un 'progetto' che vi sarà assegnato o che proporrete voi
  - orale su tutto quello discusso a lezione e su quanto fatto in laboratorio

# Slides & libri

link per i files:

- <http://www.fisgeo.unipg.it/~duranti/Sito/home.html>

(per ora trovate solo la roba del 2017-2018)

libri suggeriti:

- W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Third Edition (Cambridge University Press, 2007, ISBN-10: 0521880688)
- E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, (Pearson Education, 1994, ISBN: 0201633612, ISBN-13: 9780201633610)

**Unix/Linux**



# Linux

Da Wikipedia, l'enciclopedia libera.

🔗 *Disambiguazione* – Se stai cercando il kernel su cui si basano i sistemi operativi di questa famiglia, vedi **Linux (kernel)**.

**Linux** (<sup>i</sup><sup>1</sup>/ˈlɪnʊks/<sup>[1]</sup>, pronuncia inglese <sup>i</sup><sup>2</sup>[ˈlɪnɔːks]/<sup>[2]</sup>) è una famiglia di sistemi operativi di tipo **Unix-like**, pubblicati sotto varie possibili **distribuzioni**, aventi la caratteristica comune di utilizzare come **nucleo** il kernel **Linux**.

Oggi molte società importanti nel campo dell'**informatica** come **Google**, **IBM**, **Oracle Corporation**, **Hewlett-Packard**, **Red Hat**, **Canonical**, **Novell** e **Valve** hanno infatti sviluppato e pubblicato, e continuano a farlo, sistemi Linux.



Tux, la mascotte del kernel Linux, nata mediante uno scambio di e-mail in una **mailing list** pubblica

## Linux

- sistema operativo unix-like
- open-source (licenza GPL)
- sviluppato da Linus Torvalds come “hobby” e diffuso su Internet il 17 settembre 1991
- unico kernel (“cuore”) ma diverse distribuzioni



## Annuncio sul newsgroup:

```
Hello everybody out there using minix -
I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready.I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system(due to practical reasons)
among other things). I've currently ported bash(1.08) and gcc(1.40),and
things seem to work.This implies that I'll get something practical within a
few months, andI'd like to know what features most people would want. Any
suggestions are welcome, but I won't promise I'll implement them :-)
```

Linus (torvalds@kruuna.helsinki.fi)

```
PS. Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's
all I have :-).
```

# Linux

Come Windows e praticamente tutti i sistemi operativi, anche Linux organizza i files in una struttura ad albero

- la directory *radice*, in cima, è indicata con “/”;
- ogni directory (dir) può contenere altre directory o files;
- le sottocartelle si indicano col nome preceduto da “/”;
- se sulla macchina esiste l’utente *duranti*, la sua “home” sarà in */home/duranti* (o anche *~/duranti*);
- se l’utente *duranti* inserisce il file *LEGGIMI.txt* nella sua “home” allora il percorso (*path*) assoluto del file sarà *~/duranti/LEGGIMI.txt*  
se mi trovo nella cartella */home/duranti*, si può specificare anche il solo path relativo *./LEGGIMI.txt*  
(la cartella attuale è identificata da *./*, quella sopra da *../*)

# Linux

```
bin
boot
├── grub
├── cdrom
├── dev
├── ...
├── etc
│   ├── acpi
│   ├── ...
│   ├── cron.d
│   ├── cron.daily
│   ├── init.d
│   ├── ld.so.conf.d
│   ├── modprobe.d
│   ├── rc...d
│   ├── skel
│   ├── sudoers.d
│   ├── X11
│   └── ...
├── home
│   ├── lost+found
│   └── marcello
├── lib
│   ├── modprobe.d
│   ├── modules
│   └── ...
├── lib64
├── lost+found [error opening dir]
├── mnt
├── opt
├── proc
│   ├── acpi
│   └── ...
├── root [error opening dir]
└── run
```

## Per gestire le directory (cartelle)

cd	cambia directory
ls	mostra i file e le sottodirectory
mkdir	crea una directory
rmdir	cancella una directory vuota
pwd	stampa a schermo la directory corrente
cd .	non fa nulla
cd ..	passa nella directory superiore

rm -Rf se piena

## Per gestire i file (documenti)

touch	crea un file
more e less	visualizzano, un po' alla volta, un file di testo della directory corrente
mkdir	crea un directory
rmdir	cancella un directory vuota
rm	cancella un file
cp	copia uno o più file
mv	sposta uno o più file
grep	trova del testo all'interno di un file

\*se il file o la dir è sullo stesso file system (*fs*), *mv* semplicemente lo "rinomina" cambiandogli il path

## Per gestire i processi

top	vede i processi attivi e il consumo di CPU
ps	vede i processi dell'utente
kill	uccide un processo

## Per ridirigere input e output

< filename	leggi l'input dal file "filename"
> filename	scrivi l'output in "filename" (cancellando tutto il resto del file)
>> filename	appendi l'output alla fine di "filename"

# Linux

- **cd** senza argomenti riporta nella home directory, quella in cui ci si trova dopo essersi collegati; per l'utente Topolino questa è /home/Topolino.
  - **cd ..** sposta nella directory superiore;
  - **cd dir** sposta nella directory di percorso relativo *dir*;
  - **cd /dir** sposta nella directory di percorso assoluto *dir*.
- 
- **rm** cancella un file solo se si ha il permesso di cancellarlo.
- 
- **more e less** devono essere seguiti dal nome del file da visualizzare, che deve essere rigorosamente un file di testo. Esempio: more pippo.dat, less main.c.
- 
- **cp file1 dir1** fa una copia del file1 nella directory dir1.

# Linux

- **cp file1 file2** fa una copia di file1 di nome file2 nella stessa directory. Attenzione!! file2 potrebbe già esistere e nessuno vi chiederebbe se volete proprio sovrascriverlo! se copiate più file in una directory mydir e poi vi accorgete che mydir in realtà non esiste, tutti i file saranno stati copiati in un file di nome mydir uno sull'altro: alla fine il file mydir conterrà solo una copia dell'ultimo file copiato; per evitare questo ed altri problemi usare l'opzione cp -i che chiede conferma prima di sovrascrivere.

- **mv file1 mydir** sposta file1 nella directory mydir.

- **mv file1 file2** rinomina file1 file2. Valgono in questo caso i problemi segnalati per cp con l'aggravante che i file vengono anche cancellati dalla posizione originaria, e quindi se mydir non esiste potreste proprio prederli! Anche qui esiste l'opzione mv -i che è caldamente raccomandata!

# Linux

Nei nomi di file \* indica una qualunque sequenza di caratteri: quindi `ls *.cpp` elencherà tutti i file il cui nome finisce per ".cpp". Invece `[aAx]` indica uno dei caratteri 'a', 'A' e 'x'. il nome del file `[bB]*.txt` indica tutti i file che cominciano per 'b' oppure 'B', finiscono per '.txt' e contengono in mezzo qualunque tipo e numero di caratteri. `[a-z]` indica tutte le lettere minuscole, `[0-9]` tutte le cifre, e così via. Infine il carattere "?" sostituisce esattamente un carattere.

Domanda: trovo in una mia cartella il file \*: è il caso che lo cancelli con "rm \*" ?

# Linux

**man** Si può sapere tutto su un comando del sistema operativo linux con il comando **man**. Ad esempio **man ls** mostra tutte le opzioni per avere il listato di una cartella e **man grep** mostra le molte possibilità del comando grep.

**info** Sono un tentativo del progetto GNU di scrivere una documentazione in forma ipertestuale in tempi in cui internet e il linguaggio HTML non esistevano. Sono piuttosto complicate da navigare, ma per fortuna oggi è possibile visualizzarle graficamente. Il modo testuale inizia comunque col comando

*info comando*

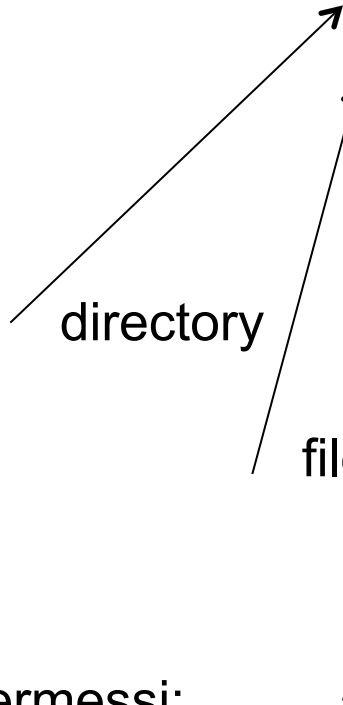


# Linux

- **cat file.txt** fa scorrere il file sullo schermo, **cat file1.txt > file2.txt** sovrascrive file2.txt col contenuto di file1.txt, **cat file1.txt >> file2.txt** appende il contenuto di file1.txt alla fine di file2.txt;
- **date** mostra data e ora, **date +%Y** mostra l'anno a 4 cifre **date +%y** mostra l'anno a 2 cifre, **date +%x** mostra solo la data **date +%X** mostra solo l'ora
- **cal mese anno** mostra il calendario, **cal anno** quello di tutto l'anno;
- **bc** è un piccolo calcolatore. Introduce in una shell, da cui si esce digitando "quit", che permette di fare le operazioni +,-,\*,/,sqrt(), ^ (potenza)

# Linux (ls)

```
bozzo@MacBook-Pro-di-Matteo-Duranti:DAQ> ls -Flarth
drwxr-xr-x  2 bozzo staff  476B 27 Giu 2015 SlowControl/
-rwxr-xr-x  1 bozzo staff  46K 27 Giu 2015 file.dat*
bozzo@MacBook-Pro-di-Matteo-Duranti:DAQ>
```



permessi:

- ogni tripletto si riferisce *u* (*user*), *g* (*group*) o *o* (*other*)
- all'interno del tripletto '-' indica che il rispettivo utente NON ha il permesso, mentre una lettera che ce l'ha: *r* (*read*), *w* (*write*) o *x* (*execute*)

# Linux

- **chmod file** agisce sui permessi dei file che sono lettura (r), scrittura (w) ed esecuzione (x), e possono essere applicati all'utente (u), al gruppo (g) o a tutti gli altri (o). I permessi possono essere aggiunti (+) tolti (-) o assegnati (=). Esiste anche un modo piú compatto per assegnare i permessi usando i numeri;
- **chown** cambia il proprietario di un file: **chown marcello file** assegna file all'utente marcello. L'opzione -R applica ricorsivamente chown se invece di un file agisco su di una directory
- **chgroup** fa lo stesso cambiando gruppo. Si puó anche usare **chown utente:gruppo file** per cambiare le due cose insieme

# Linux

- **cut -c 20-30 file.txt** seleziona i caratteri delle colonne da 20 a 30 nel file
- **head -n file.txt** mostra le prime n righe (10 se ometto n)
- **tail -n file.txt** fa lo stesso con le ultime n
- **diff file1.txt file2.txt** mostra la differenza tra due file. Si può anche usare con due cartelle
- **df** o **df -h** mostra l'occupazione di spazio su disco
- **file miofile** mostra che tipo di file è miofile (di test, eseguibile, etc.)

# Linux

- **tar** serve a creare un unico file da più file per archivarli (è acronimo di tape archive).  
**tar cvf archivio.tar file1 file2 ...** unisce file1, file2... in un unico file archivio.tar. Questo può poi essere compresso, cosa che si può fare anche con **tar cvfz archivio.tar file1 file2 ...**. Il contenuto dell'archivio si vede poi con **tar tvf (o tzvf) archivio.tar** o espanso nuovamente con **tar xvf (o xzvf) archivio.tar**
- **gzip file / gunzip file.gz** servono a comprimere / decomprimere un file
- **find** serve a trovare file con caratteristiche particolari. Ad esempio *find . -name \*.jpg* trova tutti i file jpg sotto la cartella corrente, *find ~ -size +1000000* trova tutti i file sotto la cartella home che hanno dimensione maggiore di un megabyte, *find . -mtime +2* trova tutti i file modificati da più di due giorni.

# Linux

- **sort** serve ad ordinare le linee di un file di testo
- **uniq** serve a omettere (o riportare) linee ripetute
- **sed** serve a 'editare' uno stream (i.e. un file di testo, ma successivamente vedremo che può essere usato anche con il risultato di un altro comando o programma)

```
sed 's/test/example/g' myfile.txt > newfile.txt
```

cerca la parola *test* in *myfile.txt* e rimpiazza ogni sua occorrenza con la parola *example*

- **wc** serve a mostrare quante linee (delimitate da 'a capo'), parole (delimitate da uno 'spazio bianco') e byte contiene uno o più file

# Pipe (|)

Comando 'general purpose' per "concatenare" processi e comandi

Simile a ">", ma molto più generale.

Utile per 'chainare' comandi, programmi, etc..., fra di loro:

```
cat LEGGIMI.txt | grep "pippo"
```

- *cat* 'printa' il contenuto del file LEGGIMI.txt
- *grep "pippo"* 'printa' solo le righe che contengono la sequenza di caratteri 'pippo'

Se vogliamo ordinare i contenuti di tutti i file di testo (estensione ".txt") presenti in quella directory, rimuovendo i duplicati (anche con caso differente) e salvando il risultato sul file di testo "result-file", come possiamo fare?

# Pipe (|)

Comando 'general purpose' per "concatenare" processi e comandi

Simile a ">", ma molto più generale.

Utile per 'chainare' comandi, programmi, etc..., fra di loro:

```
cat *.txt | sort -f | uniq -i > result-file
```

Ordina (case insensitive) i contenuti di tutti i file di testo (estensione ".txt") presenti in quella directory, rimuovendo i duplicati (case insensitive) e salvando il risultato sul file di testi "result-file"



# Foreground/background

- un comando lanciato dal terminale “girerà” in *foreground* (i.e. il terminale rimane “occupato” finchè il comando non ha finito)
- se si vuole far girare in *background* (i.e. il comando continua a girare ma il terminale ci ritorna ad essere utilizzabile) basta aggiungere un & alla fine della riga

```
rm -Rf /path/to/dir &
```

- un comando lanciato in *foreground* può essere messo in *background* sospendendolo (CTRL-Z, ci ritorna il terminale) e poi eseguendo il comando *bg*

# Linux

- **awk** è un filtro generico in grado di processare e manipolare, con un linguaggio simile al C, file di testo. Può essere usato creando dei veri script oppure in una *pipeline* (|)

```
cat /etc/passwd | awk 'BEGIN { FS=":" } ($2 == "**") { print $3 " - " $1 }'
```

printa il contenuto di '/etc/passwd' e lo passa ad *awk* che

- setta il Field Separator (FS) al carattere ":" (invece che lo spazio bianco), all'inizio ('BEGIN { }');
- fa un "if" sulla seconda "colonna", riga per riga, confrontandola con il carattere "\*". Questo agisce da filtro;
- di quelli "filtrati", printa la prima e la terza "colonna", inframmezzati da " – " (spazio+'-' +spazio)

# Variabili

- in generale sono identificate da un \$ all'inizio del nome
- in bash per definire una variabile:  
*export PIPPO=5*  
*export PLUTO="prova"*
- variabili "interne":
  - *\$PWD* il percorso della dir corrente
  - *\$OLDPWD* il percorso della dir precedente
  - *\$BASH* il percorso della bash stessa
  - *\$HOME* il percorso della propria home
  - *\$?* exit-status dell'ultimo comando/processo eseguito
  - ...

# Apici

- una variabile fra doppi apici (“”) viene vista come un’unica parola (anche se all’interno c’è uno spazio bianco) e in generale previene che caratteri speciali siano re-interpretati:

```
$> export PIPPO=3  
$> echo "$PIPPO"  
3
```

- una variabile fra apici singoli (‘’) non viene affatto re-intepretata:

```
$> export PIPPO=3  
$> echo '$PIPPO'  
$PIPPO
```

- *un comando fra apici storti (`) viene eseguito e il risultato (std output) trattato come una variabile (più parole)*

```
$> echo `ls`  
JINF_00.conf SlowControl TakeData
```

# Editing file di testo

## Principali comandi di Emacs

Se usato dentro XWindow, `emacs` mostra dei menù con comandi selezionabili col mouse. Comunque, esso è completamente gestibile da tastiera. Ecco una tabella che mostra i principali comandi da tastiera di `emacs`. CTRL indica il tasto Control, che va tenuto premuto mentre si batte il tasto successivo; ESC va invece battuto separatamente; gli argomenti in corsivo devono essere finiti da un `Enter`.

Principali comandi di <code>emacs</code>	
Undo	CTRL-x u oppure CTRL-_
Salva il file	CTRL-x CTRL-s
Salva con nome diverso	CTRL-x CTRL-w <i>nome</i>
Apri un nuovo file	CTRL-x CTRL-f <i>nome</i>
Inserisce un file	CTRL-x i <i>nome</i>
Passa ad un altro buffer	CTRL-x b
Chiude un buffer	CTRL-x k
Divide la finestra in due	CTRL-x 2
Passa da una metà all'altra	CTRL-x o
Riunifica la finestra	CTRL-x 1
Refresh della finestra	CTRL-l
Quit da <code>emacs</code>	CTRL-x CTRL-c
Cursore a fine riga	CTRL-e

# Editing file di testo

Cursore a inizio riga	CTRL-a
Cursore giù una pagina	CTRL-v
Cursore su una pagina	ESC v
Inizio del buffer	ESC <
Fine del buffer	ESC >
Vai alla linea...	ESC x goto-line <i>numero</i>
Cerca testo	CTRL-s <i>testo</i>
Sostituisce testo	ESC % <i>testo1 testo2</i>
Marca inizio di un blocco	CTRL-SPACE
Marca fine blocco e taglia	CTRL-w
Marca fine blocco e copia	ALT-w
Incolla blocco	CTRL-y
Pagina di aiuto	CTRL-h CTRL-h
Significato di un tasto	CTRL-h k <i>tasto</i>
Significato di tutti i tasti	CTRL-h b
Interrompe comandi complessi	CTRL-g
Apri una shell dentro <i>emacs</i>	ESC x shell
Aiuto psicologico	ESC x doctor
Torri di Hanoi	ESC x hanoi

# Linux

Esercitazione (facoltativa):

- listare tutti i file presenti nella propria cartella *home* sul file di testo LISTA.txt
- listare tutti i file presenti nelle (prime 3) sottodirectory della propria *home* e aggiungerli al file LISTA.txt
- cercare nel file LISTA.txt le linee che si riferiscono a directory (aiuto: *grep "^d"*) e scriverli nel file LISTA\_DIR.txt
- scrivere solamente i permessi, la dimensione del file e il nome del file, di tutti i file (ma non directory) in LISTA.txt nel file LISTA\_FILE\_STRIPPED.txt