

Linux Base

**La shell bash**

**Comandi base**

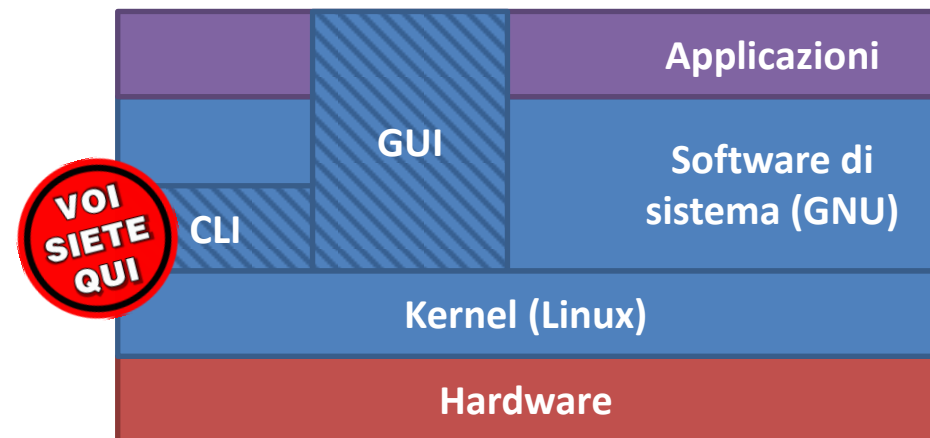
**Comandi avanzati**

**Espressioni regolari**

```
giovanni@VirtualJack:~$ echo "Real men and women don't need GUIs"  
Real men and women don't need GUIs
```

# Motivazioni

- Accesso (più) diretto al sistema
- Sistemi remoti (ssh)
- Operazioni ripetute (batch file)
- Esecuzione automatica

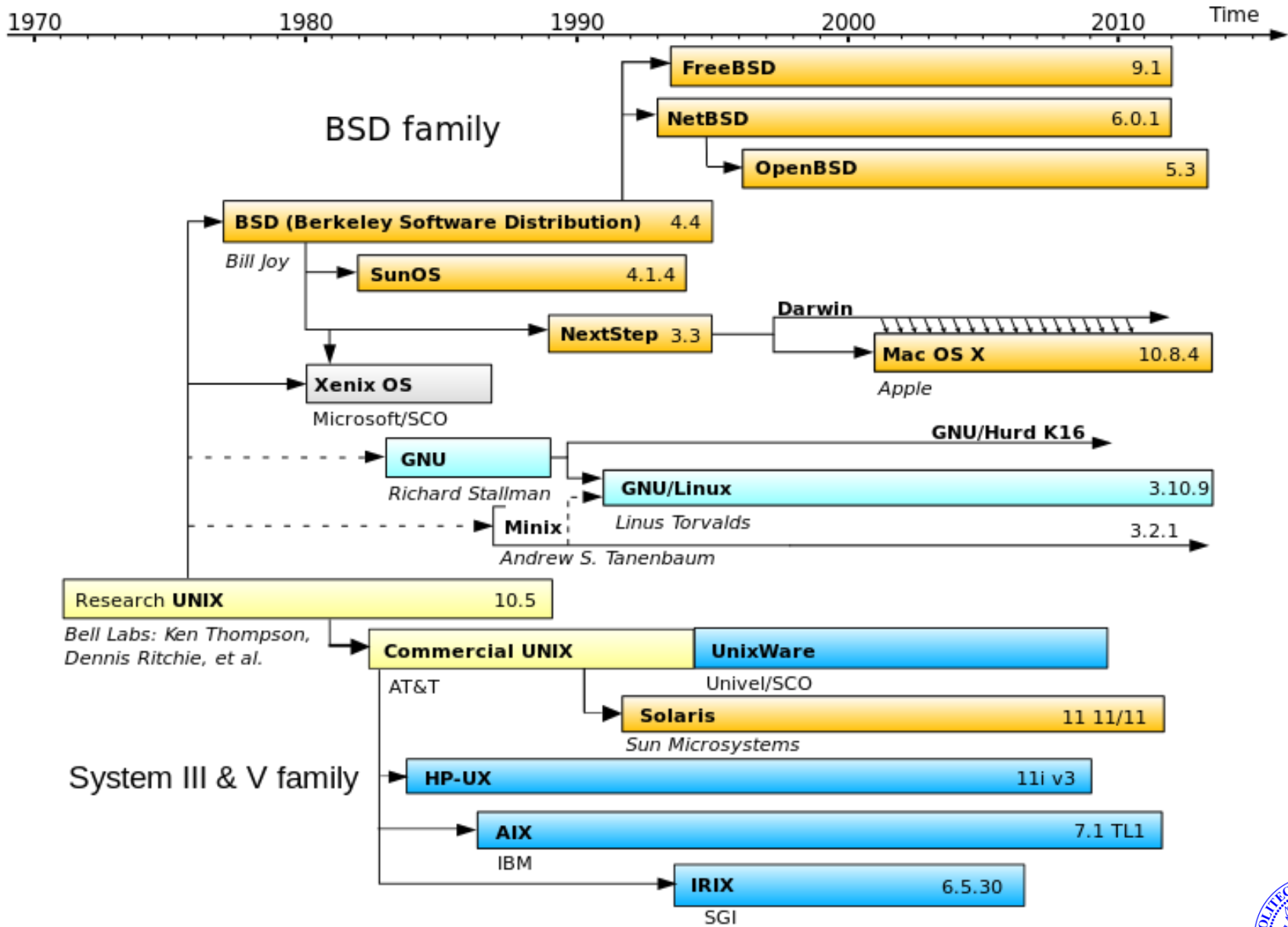


# BASH

- Standard (\*NIX, MacOS, Windows, ...)
- Versatile e potente
- Programmabile

```
MacBook-Pro-Pawe:~$ pwd
/opt/local/etc/bash_completion.d
MacBook-Pro-Pawe:~$ ls
git-completion.bash
MacBook-Pro-Pawe:~$ head -10 git-completion.bash
# bash/zsh completion support for core Git.
#
# Copyright (C) 2006,2007 Shawn O. Pearce <spearce@spearce.org>
# Conceptually based on gitcompletion (http://gitweb.hawaga.org.uk/).
# Distributed under the GNU General Public License, version 2.0.
#
# The contained completion routines provide support for completing:
#
# *) local and remote branch names
# *) local and remote tag names
MacBook-Pro-Pawe:~$
```





# Lavorare con la Console (inizio)

- Control+Alt+F1
- Usare l'applicazione «Terminale»



# Lavorare con la Console (fine)

- Alt+F7
- Comando «Exit»
- Control+D
- Chiudere il terminale

# Convenzioni

- Alcuni comandi richiedono *argomenti*

```
cd NOME_CARETELLA
```

- Il comportamento di (quasi) tutti i programmi può essere modificato attraverso *opzioni*

```
cat -n foo.txt
```

- Opzioni corte vs. opzioni lunghe

```
cat -ns foo.txt
```

```
cat --number --squeeze-blank foo.txt
```



# Comandi

- Interni (builtin commands)
- Esterni
- Alias



# Aiuto / Informazioni Aggiuntive

- Documentazione in `/usr/share/doc`
  - E.g.: `less /usr/share/doc/bash/INTRO.gz`





# Aiuto / Informazioni Aggiuntive

- Informazioni su di un comando

`comando --help` (oppure: `comando -h`)

`man comando` (e.g.: `man man`)

`whatis comando`

`apropos comando`

`type [-a] comando`

`which comando`

`info [ comando/opzioni ]`

# Il Prompt

- Il prompt standard mostra
  - nome utente «@» nome macchina «:»  
cartella corrente «\$»  
(ovvero «#» se l'utente è l'amministratore, ed è possibile fare molti danni)

```
giovanni@VirtualBob:/usr/share/doc/bash$ ls
changelog.Debian.gz  inputrc.arrows  README.abs-guide
COMPAT.gz           INTRO.gz        README.bash_completion.gz
copyright           NEWS.gz         README.commands.gz
FAQ.gz             POSIX.gz        README.Debian.gz
giovanni@VirtualBob:/usr/share/doc/bash$ █
```

# Cartelle speciali

- Punto «.»
  - la cartella corrente
- Punti-punto «..»
  - la cartella padre della corrente
- Tilde «~»
  - la propria cartella *home* (di solito /home/utente)
- Tilde + nome «~foo»
  - la cartella *home* di foo (di solito /home/foo)

# Edit della linea di comando

- Bash permette di usare le frecce del cursore, e combinazioni di tasti usate da Emacs
  - Control+A, Control+E, Meta+f, Meta+b, ...
- «Freccia su»/«Freccia giù» permettono di muoversi fra i comandi precedenti

# Autocompletamento



# Control

- Control+C
  - Invia SIGINT al processo corrente
- Control+D
  - End Of File
  - Equivalente a Control+Z in MS-DOS/Windows



# Variabili

- La shell permette di definire «variabili»
- Dollaro «\$» per usare le variabili definite
- Doppi apici vs. apice singolo

```
giovanni@VirtualJack:~$ FOO=bar
giovanni@VirtualJack:~$ echo FOO
FOO
giovanni@VirtualJack:~$ echo $FOO
bar
giovanni@VirtualJack:~$ echo "$FOO"
bar
giovanni@VirtualJack:~$ echo '$FOO'
$FOO
```

# Variabili

- Il comportamento e le opzioni di default della shell e di numerosi comandi sono definiti attraverso variabili
- L'insieme delle variabili è detto «ambiente» (*environment*)
- Usare il comando «set» per visualizzare le circa 50 variabili definite nell'ambiente corrente

# \$PATH

- I comandi esterni vengono cercati nelle cartelle specificate nella variabile PATH
- È possibile specificare in modo esplicito il path dei comandi (relativo o assoluto)

```
giovanni@VirtualJack:/$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
giovanni@VirtualJack:/$ which echo
/bin/echo
giovanni@VirtualJack:/$ echo CIAO
CIAO
giovanni@VirtualJack:/$ /bin/echo CIAO
CIAO
giovanni@VirtualJack:/$
```

# Alias

- È possibile creare «**alias**» ai comandi (e.g., per memorizzare le opzioni usate sempre)

```

giovanni@VirtualJack:~$ alias zop='ls -Al'
giovanni@VirtualJack:~$ zop
totale 28
-rw----- 1 giovanni giovanni 3675 ago 25 14:26 .bash_history
-rw-r--r-- 1 giovanni giovanni 220 ago 24 08:47 .bash_logout
-rw-r--r-- 1 giovanni giovanni 3486 ago 24 08:47 .bashrc
drwx----- 2 giovanni giovanni 4096 ago 24 08:47 .cache
-rw-rw-r-- 1 giovanni giovanni 0 ago 25 12:15 Hey Joe
drwxrwxr-x 2 giovanni giovanni 4096 ago 24 16:09 .landscape
-rw----- 1 giovanni giovanni 51 ago 25 14:13 .lessht
-rw-r--r-- 1 giovanni giovanni 675 ago 24 08:47 .profile
giovanni@VirtualJack:~$ alias zop
alias zop='ls -Al'

```

- «**alias**» senza parametri mostra tutti gli alias

# Nomi dei file

- I sistemi POSIX mettono poche restrizioni ai nomi dei file, tuttavia è meglio evitare:
- spazi, tab e «a capo»
- caratteri che hanno significati convenzionali

/ \ " ' \* ; ? [ ] ( )  
 ~ ! \$ { } < > # @ & |

- lettere al di fuori dal set ASCII base

e.g.: ω è ë ê æ ...

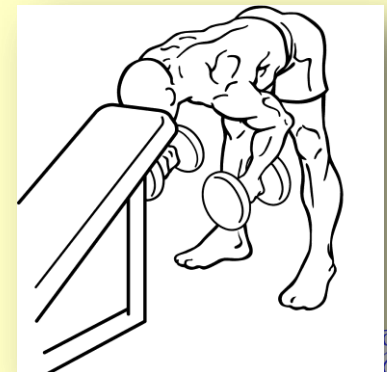
# Nomi dei file

- I nomi che contengono spazi devono essere racchiusi da apici o doppi apici

```
giovanni@VirtualJack:~$ touch "Hey Joe"  
giovanni@VirtualJack:~$ ls  
Hey Joe  
giovanni@VirtualJack:~$
```

# Esercizio

- A cosa servono i comandi «**basename**» e «**dirname**»?
  - usare il manuale
  - fare alcune prove



# Contenuto delle cartelle

**ls [-laAs] [ f1 ... fN ]**

- mostra contenuto di una cartella
- Nota: in Ubuntu 12.04 il comando ha 44 opzioni diverse

```

giovanni@VirtualBob:~$ ls -al /tmp/
totale 40
drwxrwxrwt  9 root      root      4096 Aug 25 12:39 .
drwxr-xr-x 23 root      root      4096 Aug 24 08:45 ..
drwx-----  2 root      root      4096 Aug 25 12:38 aptitude-root.4192:Iu4kUm
drwxrwxrwt  2 root      root      4096 Aug 25 10:42 .ICE-unix
drwx-----  2 giovanni giovanni 4096 Aug 25 10:42 keyring-JjdyIh
drwx-----  2 giovanni giovanni 4096 Aug 25 10:42 pulse-7pHQdUGW26bZ
drwx-----  2 root      root      4096 Aug 25 10:42 pulse-PKdhtXMmr18n
drwx-----  2 giovanni giovanni 4096 Aug 25 10:42 ssh-CviYHJWw1248
-r--r--r--  1 root      root           11 Aug 25 10:42 .X0-lock
drwxrwxrwt  2 root      root      4096 Aug 25 10:42 .X11-unix
  
```



# Significato del «formato lungo»

```
drwx----- 2 giovanni giovanni 4096 Aug 25 10:42 ssh-CviYHJWw1248
-r--r--r-- 1 root      root      11 Aug 25 10:42 .X0-lock
```

tipo

permessi  
user group world

gruppo

ultima  
modifica

anche i colori  
aiutano a  
distinguere i  
vari tipi

numero link

possessore

dimensione

nome

# Tipo dei file

**file f1 [ ... fN ]**

- determina il tipo di file utilizzando vari metodi: informazioni del filesystem; magia; riconoscimento elementi tipici dei linguaggi
- **man magic** per maggiori informazioni sulla magia (e sui *magic number*)



# Wildcards («*globbing*»)

- Wildcards: \* ? [ ] -
- A differenza di MS-DOS/Windows l'espansione è fatta dalla shell

```

giovanni@VirtualJack:/$ ls
bin  etc      initrd.img.old  media  proc  sbin  sys  var
boot home    lib             mnt    root  selinux tmp  vmlinuz
dev  initrd.img lost+found      opt    run   srv   usr  vmlinuz.old
giovanni@VirtualJack:/$ echo *
bin boot dev etc home initrd.img initrd.img.old lib lost+found media mnt opt pro
c root run sbin selinux srv sys tmp usr var vmlinuz vmlinuz.old
giovanni@VirtualJack:/$ echo ???
bin dev etc lib mnt opt run srv sys tmp usr var
giovanni@VirtualJack:/$ echo *[aeiou]
home media
giovanni@VirtualJack:/$ echo s[a-k]*
sbin selinux

```

# Comandi precedenti

## `history`

- mostra i comandi precedenti
- Nota: è possibile usare e modificare i vecchi comandi con sequenze che iniziano con «!» e «^»
  - E.g.: `!-2:s/foo/bar/g`
  - `man history`

# Navigazione

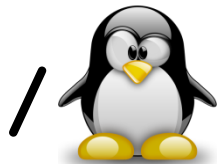
**cd [dir]**

- cambia cartella

**pwd**

- mostra la cartella corrente

# Esempio



/

/tmp/

/tmp/dir1/

/tmp/dir1/file.txt

/tmp/dir2/

/tmp/dir2/file.txt

```
/$ cd /tmp/dir1
```


# Esempio

```
/$ cd ../dir2
```

/

/tmp/

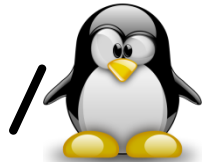
/tmp/dir1/ 

/tmp/dir1/file.txt  


/tmp/dir2/ 

/tmp/dir2/file.txt

# Esempio



/tmp/

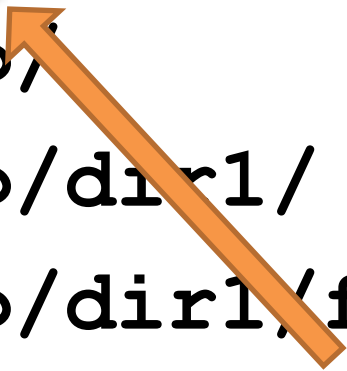
/tmp/dir1/

/tmp/dir1/file.txt

/tmp/dir2/ 

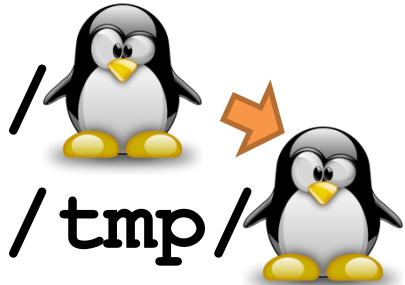
/tmp/dir2/file.txt

```
/$ cd ../..
```





# Esempio



/tmp/dir1/

/tmp/dir1/file.txt

/tmp/dir2/

/tmp/dir2/file.txt

```
/$ cd /tmp/dir1/../../dir2/..
```

# Creazione

**touch f1 [ ... fN ]**

– *tocca* (crea o aggiorna l'istante dell'ultima modifica)

**mkdir [-p] dir**

– crea una directory

# Copia/Collegamento

```
cp [-pr] f1 f2
```

```
cp [-pr] f1 [ ... fN ] dir
```

– copia file

```
ln [-s] f1 [ f2 ]
```

```
ln [-s] f1 [ ... fN ] dir
```

– crea un collegamento

– nessun hard link a cartella, o fra filesystem diversi

– i dati sono cancellati se non più *linkati* da nessun file

# Cancellazione

```
rm [-rf] f1 [ ... fN ]
```

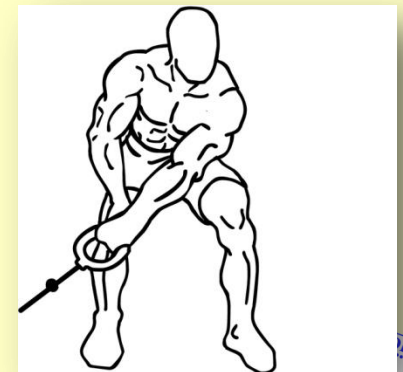
- rimuove file
- alias comune: « **alias rm='rm -I'** »

```
rmdir dir
```

- rimuove una cartella vuota

# Esercizi

- Creare un file `~/foo/uno.txt`
  - creare un link simbolico `~/bar/due.txt`
  - creare un link `~/baz/tre.txt`
- Modificare uno dei file ed osservare i cambiamenti negli altri due
- Rimuovere `uno.txt`
  - cosa succede a `due.txt` e `tre.txt`



# Rinominare/Spostare

```
mv f1 f2
```

```
mv f1 [ ... fN ] dir
```

– muove (rinomina o sposta) file

# Visualizzare

**cat** [-ns] [ f1 ... fN ]

- concatena e mostra file

**less** [ f1 ... fN ]

- mostra file (interattivo)
- usare «h» per un aiuto su comandi ed opzioni dentro less

**reset**

- (re-)inizializza il terminale

# Visualizzare

```
head f1 [ ... fN ]
```

- mostra le prime 10 righe dei file

```
tail f1 [ ... fN ]
```

- mostra le ultime 10 righe dei file

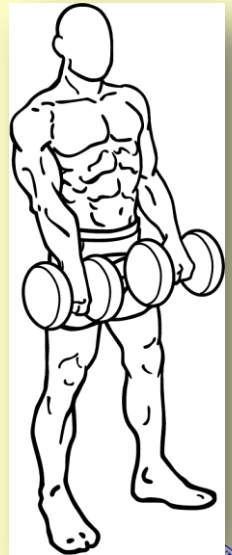
```
sort [-nu] [ f1 ... fN ]
```

- visualizza i file dopo aver ordinato le righe



# Esercizi

- Quali sono i tipi di filesystem disponibili sul vostro sistema?
  - Tip: esamina il file **filesystems** in **/proc**
- Quali sono le caratteristiche della CPU?
  - Tip: esamina il file **cpuinfo** e **meminfo** in **/proc**
- Quali sono gli utenti?
  - Tip: ogni utente ha una cartella in **/home**



# Output base

**echo** [-n] [ testo ]

- stampa il testo a video

**clear**

- cancella lo schermo
- equivalente a «Control+L»

# Informazione sull'utente

**who [am I]**

- chi sono gli utenti attivi nel sistema
- chi sono io?

**groups**

- gruppi a cui appartiene l'utente

# Modificare permessi

```
chmod [-R] modo [ f1 ... fn ]
```

- cambia permessi
- chi: **u g o a**
- modifica: **+ - =**
- permessi: **r w x X**
- speciali: **s S t**

# Modificare permessi

**chown** *utente* [:*gruppo*] *f1* [ ... *fn* ]

– cambia owner

**chgrp** *gruppo* *f1* [ ... *fn* ]

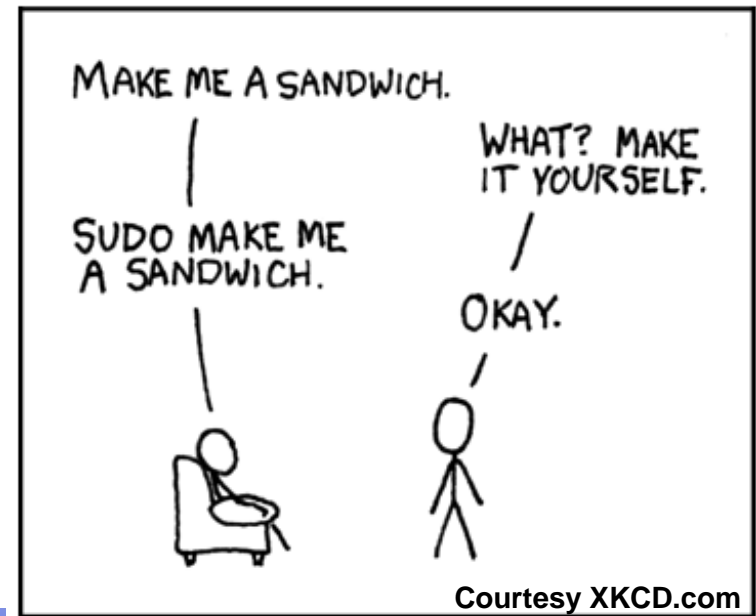
– cambia gruppo

# Root

- L'utente *root* è l'amministratore del sistema
  - Nessuna limitazione
  - Potenzialmente molto rischioso

## **sudo comando**

- esegue il singolo comando con i privilegi di root





"Spider-Man"  
Julius Fuentelba

sudo

*“With great power there must also come great responsibility”*

— Stan Lee

Immagine di JuliusC1224 (<http://juliusc1224.deviantart.com/art/Spider-Man-331969812>)

# Informazioni

**uname [-a]**

- informazioni sul sistema

**lsb\_release [-a]**

- informazioni sulla release

- File e cartelle in **/proc**



# Dimensione

**wc [f1 ... fN]**

- dimensione dei file in righe, parole, caratteri

**du [dir]**

- dimensione dei file di una cartella

# Mountpoint

## `mount`

- visualizza i filesystem montati
- per montare nuovi occorrono i privilegi di root

# Eseguire comandi

- Quando si esegue un comando la shell
  - crea un nuovo processo (fork) in cui esegue il comando
  - attende finché il nuovo processo non termina
- specificando «**exec comando**» il comando viene eseguito nel processo corrente
  - quando il comando termina, la shell ormai non esiste più
  - «**exec echo addio**» è un modo per terminare la shell

# Sleep/Date

## `sleep sec`

- interrompe l'esecuzione fino all'istante ora corrente + **sec** secondi

## `date`

- mostra la data e l'ora corrente

# Eseguire comandi

- Più comandi possono essere specificati sulla stessa linea separati con «;»

```
giovanni@VirtualJack:/$ date; sleep 5; date  
lun 25 ago 2014, 14.06.38, CEST  
lun 25 ago 2014, 14.06.43, CEST  
giovanni@VirtualJack:/$
```

# Eseguire comandi

- Comandi racchiusi fra parentesi tonde «( )» sono eseguiti in una subshell (sotto-shell)
  - la subshell eredita l'environment della shell madre
  - ma le modifiche sono locali

```
giovanni@VirtualJack:~$ FOO=BAR; echo $FOO; (echo $FOO); echo $FOO
BAR
BAR
BAR
giovanni@VirtualJack:~$ FOO=BAR; echo $FOO; (FOO=QUZ; echo $FOO); echo $FOO
BAR
QUZ
BAR
giovanni@VirtualJack:~$
```

# Processi e Job

- Ogni processo ha un numero
  - PID
- Ogni processo è collegato ad un utente
  - UID

# Processi e Job

- Altre informazioni
  - commandline
  - tempo
  - memoria
  - priorità
  - nice level



# Gentilezza

**nice num cmd**

- esegue un comando in modo più gentile

**renice num proc**

- rende un processo più gentile

# Controllo processi

**ps** [-lA] [-f f]

- mostra informazioni sui processi

**kill** [-9] **proc1** [ ... **procN** ]

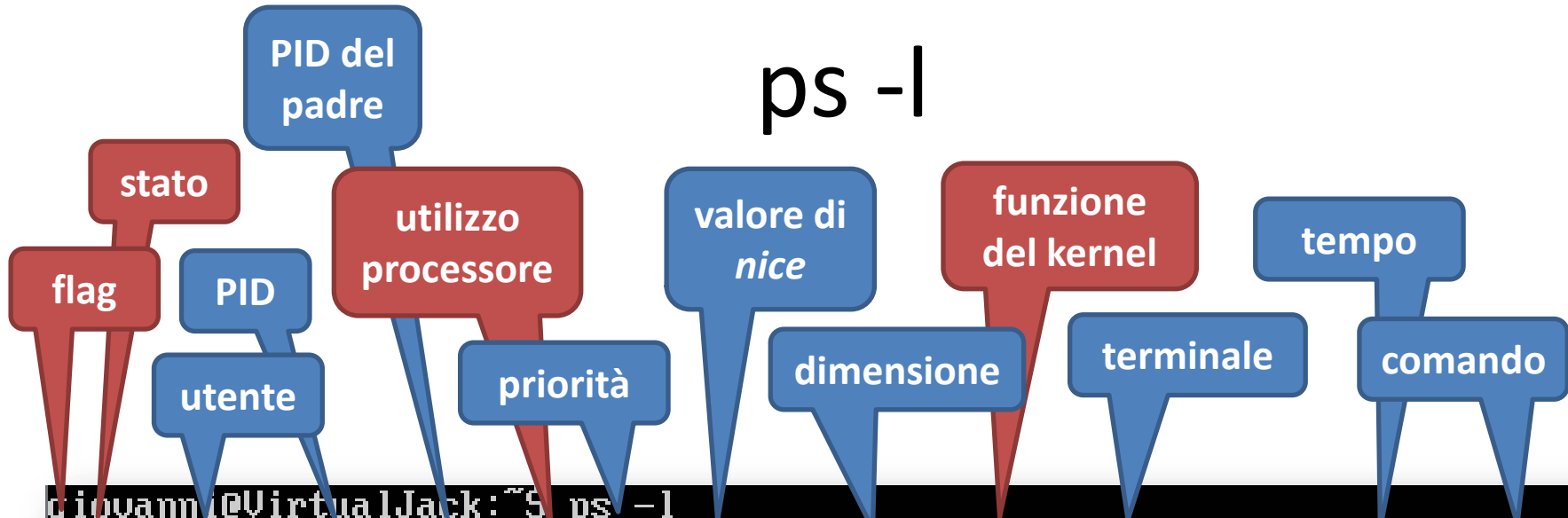
- termina i processi

**top**

**htop**

- mostra informazioni sui processi (interattivo)

# ps -l



```
giovanni@VirtualJack:~$ ps -l
F S  UID  PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  TTY  TIME CMD
4 S  1000 2917 2810  0  80   0 - 2498 wait  tty1  00:00:00 bash
0 R  1000 3316 2917  0  80   0 - 1176 -    tty1  00:00:00 ps
```

# --forest e -H

```
giovanni@moth:~$ ps --forest
  PID TTY          TIME CMD
 1024 tty1        00:00:00 bash
 1635 tty1        00:00:00  \_ ps
giovanni@moth:~$ ps -H
  PID TTY          TIME CMD
 1024 tty1        00:00:00 bash
 1636 tty1        00:00:00  ps
giovanni@moth:~$ alias ps='ps -H'
```

# Controllo job

**jobs**

- mostra informazioni sui job in esecuzioni o sospesi

**fg [%j]**

**bg [%j]**

- riprende l'esecuzione del job in primo/secondo piano

**kill [-9] %j1 [ ... %jN ]**

- termina i job

# Sospendere l'esecuzione

- L'esecuzione di un comando può essere sospesa con «Control+Z»
- L'esecuzione può essere ripresa
  - In primo piano con «fg»
  - In secondo piano (background) con «bg»

```
giovanni@VirtualJack:~$ (date; sleep 10; date)
lun 25 ago 2014, 14.22.59, CEST
lun 25 ago 2014, 14.23.09, CEST
giovanni@VirtualJack:~$ (date; sleep 10; date)
lun 25 ago 2014, 14.23.13, CEST
^Z
[1]+  Fermato          ( date; sleep 10; date )
giovanni@VirtualJack:~$ fg
( date; sleep 10; date )
lun 25 ago 2014, 14.23.39, CEST
```

# Sospendere l'esecuzione

- L'esecuzione di un comando può essere sospesa con «Ctrl-Z»
- L'esecuzione può essere ripresa
  - In primo piano con «fg»
  - In secondo piano (background) con «bg»

numero del  
job

```

giovanni@VirtualJack:~$ (date; sleep 10; date)
lun 25 ago 2014, 14.22.59, CEST
lun 25 ago 2014, 14.23.09, CEST
giovanni@VirtualJack:~$ (date; sleep 10; date)
lun 25 ago 2014, 14.23.13, CEST
^Z
[1]+  Fermato                  ( date; sleep 10; date )
giovanni@VirtualJack:~$ fg
( date; sleep 10; date )
lun 25 ago 2014, 14.23.39, CEST
    
```

# Sospendere l'esecuzione

- Specificando «&» al termine del comando questo viene eseguito subito in background

```
giovanni@VirtualJack:~$ (sleep 10; date) &
[1] 2515
giovanni@VirtualJack:~$ (sleep 10; date) &
[2] 2517
giovanni@VirtualJack:~$ (sleep 10; date)
lun 25 ago 2014, 14.27.22, CEST
lun 25 ago 2014, 14.27.24, CEST
lun 25 ago 2014, 14.27.27, CEST
[1]- Completato          ( sleep 10; date )
[2]+ Completato          ( sleep 10; date )
giovanni@VirtualJack:~$ _
```



# Controllare l'esecuzione

- Spuntando il comando «&» al termine del comando questo viene eseguito in background

job precedente

job corrente

```
giovanni@VirtualJack:~$ (sleep 10; date) &
[1] 515
giovanni@VirtualJack:~$ (sleep 10; date) &
[2] 517
giovanni@VirtualJack:~$ (sleep 10; date)
  lun 25 ago 2014, 14.27.22, CEST
  lun 25 ago 2014, 14.27.24, CEST
  lun 25 ago 2014, 14.27.27, CEST
[1]- Completato (sleep 10; date)
[2]+ Completato (sleep 10; date)
giovanni@VirtualJack:~$ _
```

# Sospendere l'esecuzione

- Il comando «**wait**» attende che tutti i processi in background siano terminati

```

giovanni@VirtualJack:~$ (sleep 30; date) &
[1] 2530
giovanni@VirtualJack:~$ (sleep 30; date) &
[2] 2532
giovanni@VirtualJack:~$ (sleep 30; date) &
[3] 2534
giovanni@VirtualJack:~$ wait
lun 25 ago 2014, 14.30.07, CEST
[1] Completato ( sleep 30; date )
lun 25 ago 2014, 14.30.08, CEST
[2]- Completato ( sleep 30; date )
lun 25 ago 2014, 14.30.09, CEST
[3]+ Completato ( sleep 30; date )
giovanni@VirtualJack:~$

```

# Albero (ascii-art)

```
giovanni@moth:~$ ( sleep 15; date ) &
[1] 1410
giovanni@moth:~$ ( sleep 30; date ) &
[2] 1412
giovanni@moth:~$ ( sleep 45; date ) &
[3] 1414
giovanni@moth:~$ ps
  PID TTY          TIME CMD
 1048 tty1      00:00:00 bash
  1410 tty1      00:00:00 \_  bash
  1411 tty1      00:00:00 |   \_  sleep
  1412 tty1      00:00:00 \_  bash
  1413 tty1      00:00:00 |   \_  sleep
  1414 tty1      00:00:00 \_  bash
  1415 tty1      00:00:00 |   \_  sleep
  1416 tty1      00:00:00 \_  ps
giovanni@moth:~$ wait
Wed Aug 27 08:05:23 CEST 2014
[1] Done ( sleep 15; date )
Wed Aug 27 08:05:43 CEST 2014
[2]- Done ( sleep 30; date )
Wed Aug 27 08:06:01 CEST 2014
[3]+ Done ( sleep 45; date )
```

# Redirezione



# Redirezione

```
$ comando parametri > file
```

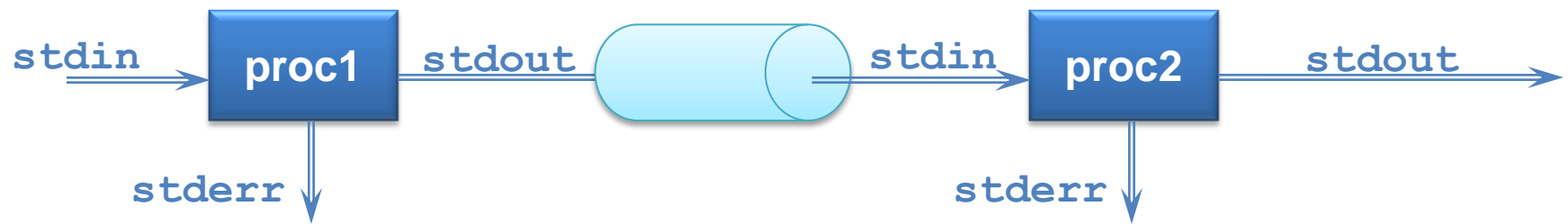
```
$ comando parametri >> file
```

```
$ comando parametri < file
```

# Redirezione (stdout vs. stderr)

```
$ comando parametri > file  
$ comando parametri > file 2> file2  
$ comando parametri > file 2>&1  
$ comando parametri >& file
```

# Pipe



# Pipe

```
$ comando1 parametri | comando2 parametri
```

```
$ comando1 parametri |& comando2 parametri
```

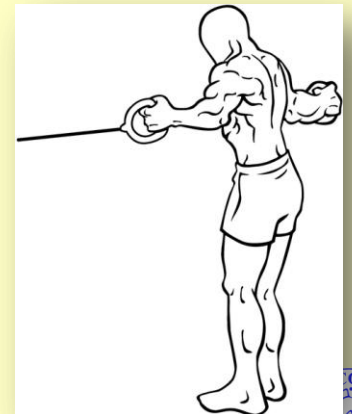


# Pipe

```
giovanni@VirtualJack:~$ cat /etc/*.conf | wc  
cat: /etc/fuse.conf: Permesso negato  
1551 5522 47736
```

# Esercizio

- Elencare tutti i file presenti in una directory in ordine di dimensione
  - usando le opzioni di **ls (man)**
  - usando il comando «**sort**»



# Redirezione

**tee [-a] file**

- duplica lo stdin salvandone una copia in *file*
- utile per salvare una copia di quello che viene visualizzato o trasmesso nella pipe

```
giovanni@VirtualJack:~$ ls -s /etc | tee hey.txt | sort -n
```

# Archivi (tarball)

- È comune distribuire materiale sotto forma di archivio (come Zip o Tar in windows)
- Tradizionalmente nei sistemi UN\*X si usano due programmi in pipe:
  - uno per trasformare più file in un unico file (**tar**)
  - uno per comprimere (**gzip**, **bzip2**, o **xzip**)

# Archivi (tar)

`tar op archivio f1 ... fN`

– operazione: `c x t f [v]`

– se il nome dell'archivio è meno «-» usa lo stdout

# Archivi (compressione)

```
prog [-9vd] [f1 ... fN]
```

```
prog [-9vd] -c
```

- comprime/decomprime file, modifica l'estensione in modo opportuno
- con l'opzione **-c** comprime/decomprime lo stdin e scrive sullo stdout
- i programmi sono: **bzip2** (molto diffuso); **gzip** (minime risorse); **xz** (nuovo)

# Archivi

```
giovanni@VirtualJack:~$ (cd /; tar cf - bin) | bzip2 -c9 > bin.tar.bz2
giovanni@VirtualJack:~$ (cd /; tar cf - bin) | gzip -c9 > bin.tar.gz
giovanni@VirtualJack:~$ (cd /; tar cf - bin) | xz -c9 > bin.tar.xz
giovanni@VirtualJack:~$ ls -l bin.tar.*
-rw-rw-r-- 1 giovanni giovanni 3346778 ago 25 16:29 bin.tar.bz2
-rw-rw-r-- 1 giovanni giovanni 3842740 ago 25 16:29 bin.tar.gz
-rw-rw-r-- 1 giovanni giovanni 2435420 ago 25 16:29 bin.tar.xz
```

# Archivi (tar)

- È possibile specificare la compressione direttamente nei parametri di **tar**
  - **z** per usare gzip
  - **j** per usare bzip2
  - **J** per usare xz



# Archivi (solo decompressione)

```
prog [-v] [f1 ... fN]
```

```
prog [-v] -c
```

- decomprime file, modifica l'estensione in modo opportuno
- con l'opzione **-c** decomprime lo stdin sullo stdout
- i programmi sono: **bunzip2**; **gunzip**; **unxz**

# Cercare file

```
find [-xdev] dir [e1 ... eN]
```

- cerca nell'albero del filesystem a partire da **dir**
- applica espressioni **e1 ... eN**
- con **xdev** non esce dal filesystem radice

# Cercare file

## Esempi

```
find ~ -name foo -print
```

```
find . -name "*foo*"
```

```
find ~ -name -type d -iname "*bar*" -ls
```

```
find . -name "*.o" -print -exec rm '{}' \;
```

```
find / -size +100M
```

```
find /usr -executable
```

```
find /etc /usr/local/etc \! -readable
```

# Cercare file

```
grep [-iv] pattern [f1 ... fN]
```

– filtra le righe che contengono un pattern

# Esercizio

- Cercare tutti i file di proprietà dell'utente corrente
  - Tip: l'opzione non è stata spiegata, usare «**man find**»



# Regex

- Una espressione regolare (*regular expression*, *regexp*, *regex*, *RE*) è una stringa che identifica un insieme di stringhe
- Operazioni di *patten matching*
- Slang: “*la stringa meccia la regex*”

# Regex: simboli base

- Un carattere
  - quel carattere
- Punto «.»
  - un carattere qualsiasi

```
ma..a
```

```
oh mamma mia!
```

```
mamma
```

# Regex: simboli base

- Cappello «^» e dollaro «\$»
  - rispettivamente, inizio e fine stringa

```
^ma..a$
```

```
eh-mamma-mia!
```

```
mamma
```



# Regex: simboli base

- Asterisco «\*»
  - un numero qualsiasi di volte, anche zero
- Più «+»
  - una o più volte
- Punto interrogativo «?»
  - zero o una volta

```
ma?m+z*ak?
```

```
oh mamma mia!
```

```
mamma
```

# Regex: simboli base

- Parentesi quadre «[ ... ]»
  - uno qualsiasi dei caratteri specificati
  - se x-y, allora uno qualsiasi dei caratteri fra x e y
  - se il primo carattere dopo «[» è «^», allora un qualsiasi carattere non specificato

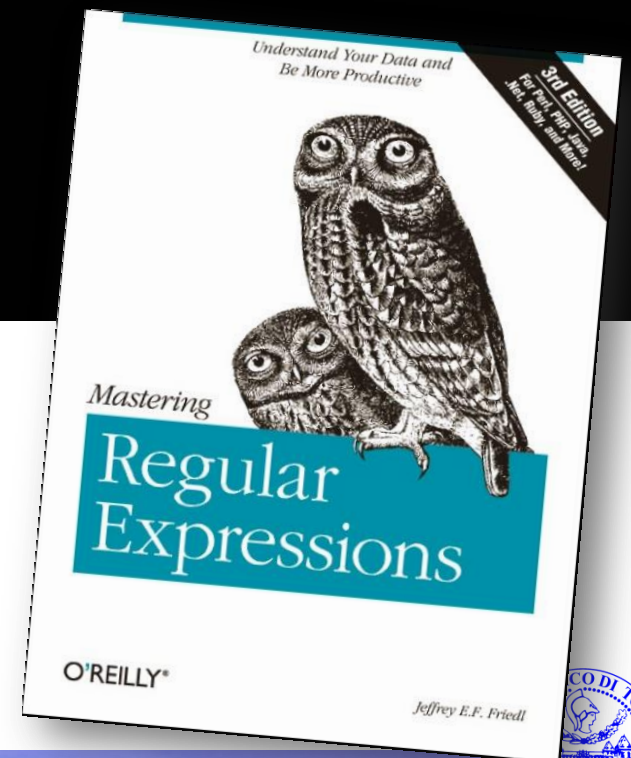
```
^[A-Z]a?o?[mn]+a$
```

**Mamma**

**Nonna**

# Regex

```
giovanni@moth:~/gx$ apropos regex
re_comp (3)      - BSD regex functions
re_exec (3)     - BSD regex functions
regcomp (3)     - POSIX regex functions
regerror (3)    - POSIX regex functions
regex (3)       - POSIX regex functions
regex (7)       - POSIX.2 regular expressions
regexexec (3)   - POSIX regex functions
regfree (3)     - POSIX regex functions
```



# Cercare file

```
grep [-iv] -e regex [f1 ... fN]
```

– filtra le righe che contengono l'espressione regolare

These slides are licensed under a **Creative Commons**

**Attribution  
Non Commercial  
Share Alike  
4.0 International**

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-nc-sa/4.0/>

Versione in Italiano:

<https://creativecommons.org/licenses/by-nc-sa/4.0/deed.it>

