Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative
Istituto sistemi informativi e networking

SUPSI

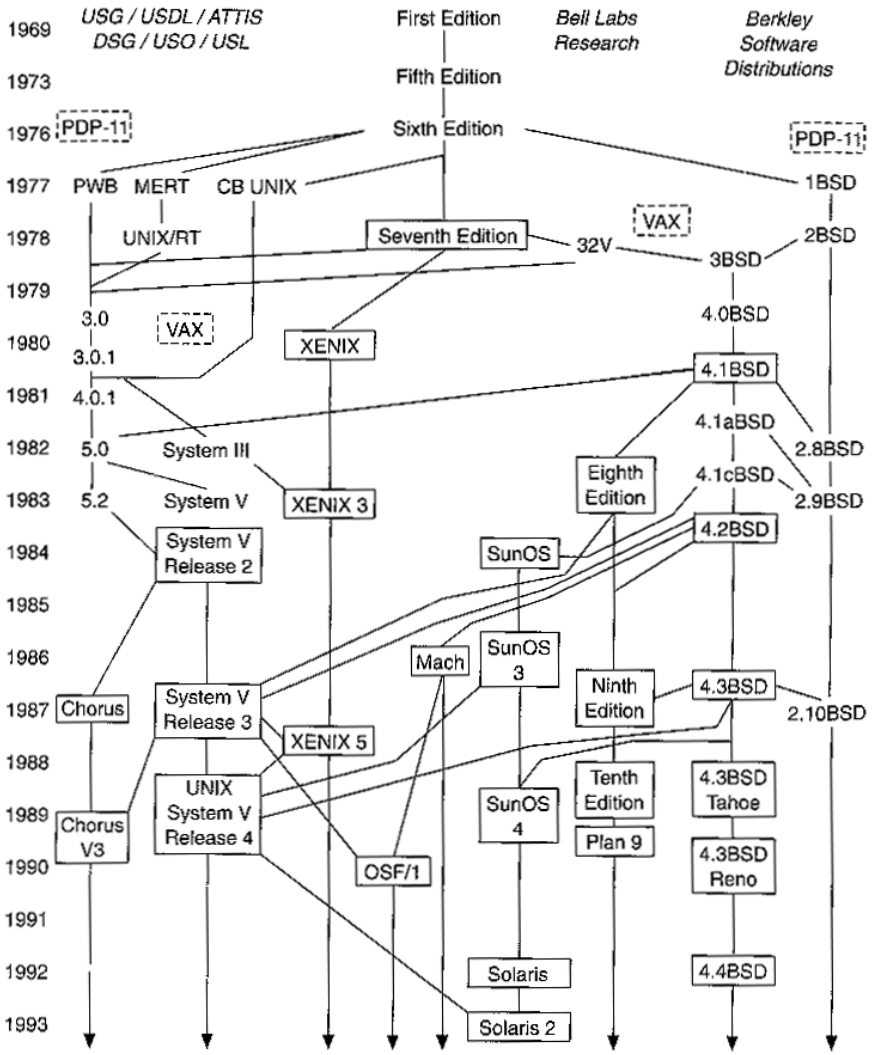
Ambienti Operativi: Bash

Amos Brocco, Ricercatore, DTI / ISIN

Unix

- Bell labs, anni 60
- Obiettivi:
  - ambiente di programmazione
  - semplice interfaccia utente
  - semplici utility che possono essere combinate per realizzare potenti funzioni
  - file system gerarchico (ad albero)
  - semplice interfacciamento con le periferiche
  - sistema multi-user e multi-tasking
- Fine anni 80: standard POSIX (**P**ortable **O**perating **S**ystem **I**nterface for Uni**x**)

Unix


1969
1973
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
USG / USDL / ATTIS
DSG / USO / USL
Bell Labs
Research
Berkley
Software
Distributions
First Edition
Fifth Edition
Sixth Edition
PDP-11
PDP-11
1BSD
PWB
MERT
CB UNIX
UNIX/RT
Seventh Edition
32V
VAX
3BSD
2BSD
3.0
VAX
XENIX
4.0BSD
3.0.1
4.1BSD
4.0.1
4.1aBSD
2.8BSD
5.0
System III
Eighth
Edition
4.1cBSD
2.9BSD
5.2
System V
XENIX 3
4.2BSD
System V
Release 2
SunOS
Mach
SunOS
3
Chorus
System V
Release 3
XENIX 5
Ninth
Edition
4.3BSD
2.10BSD
UNIX
System V
Release 4
Tenth
Edition
4.3BSD
Tahoe
Chorus
V3
SunOS
4
Plan 9
OSF/1
4.3BSD
Reno
Solaris
4.4BSD
Solaris 2

Shell Unix

- La maggior parte delle shell Unix sono programmabili, ovvero permettono l'esecuzione di semplici programmi detti script
  - Permettono di automatizzare compiti ricorrenti
  - Permettono di estendere la shell con nuove funzionalità
- Diverse shell disponibili: sh, bash, csh, tcsh, ksh,...
- Essenza della filosofia Unix
  - L'utente sa quello che fa
  - Combinare piccole funzionalità per ottenerne di più complesse

Tipi di shell

- Interattiva, non-interattiva
  - Interattiva: presenta un prompt, e accetta i comandi dell'utente
  - Non-interattiva: viene usata per eseguire degli script
- Shell di login (accesso) o no
  - Login: eseguita automaticamente dopo il login locale o remoto
  - Non login: viene eseguita esplicitamente (per es. per eseguire uno script)

  - Cambiare la shell di login

[X] bash

```
linux1.dti.supsi.ch> echo $SHELL
/bin/tcsh
linux1.dti.supsi.ch> chsh
Modifica shell per utente in corso.
Parola d'ordine:
Nuova shell [/bin/tcsh]: /bin/bash
Shell modificata.
linux1.dti.supsi.ch>
```

Bash

- Bash
  - rilasciata nel 1989
  - sviluppata da Brian Fox
  - parte del progetto GNU
  - sintassi derivata dalla Bourne shell (sh), dal nome del suo creatore; il nome Bash sta per “Bourne Again Shell”
  - Ispirata C shell, Korn shell, e altre

Login e logout

- Login: Procedura di accesso e autenticazione
  - Richiede:
    - un nome utente (che identifica la persona che vuole autenticarsi)
    - una parola chiave di accesso (password)

[X] bash

```
host login: utente
Password:
```

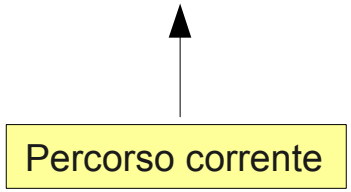
- Logout: Procedura di de-autenticazione
  - L'utente termina la sua sessione di lavoro

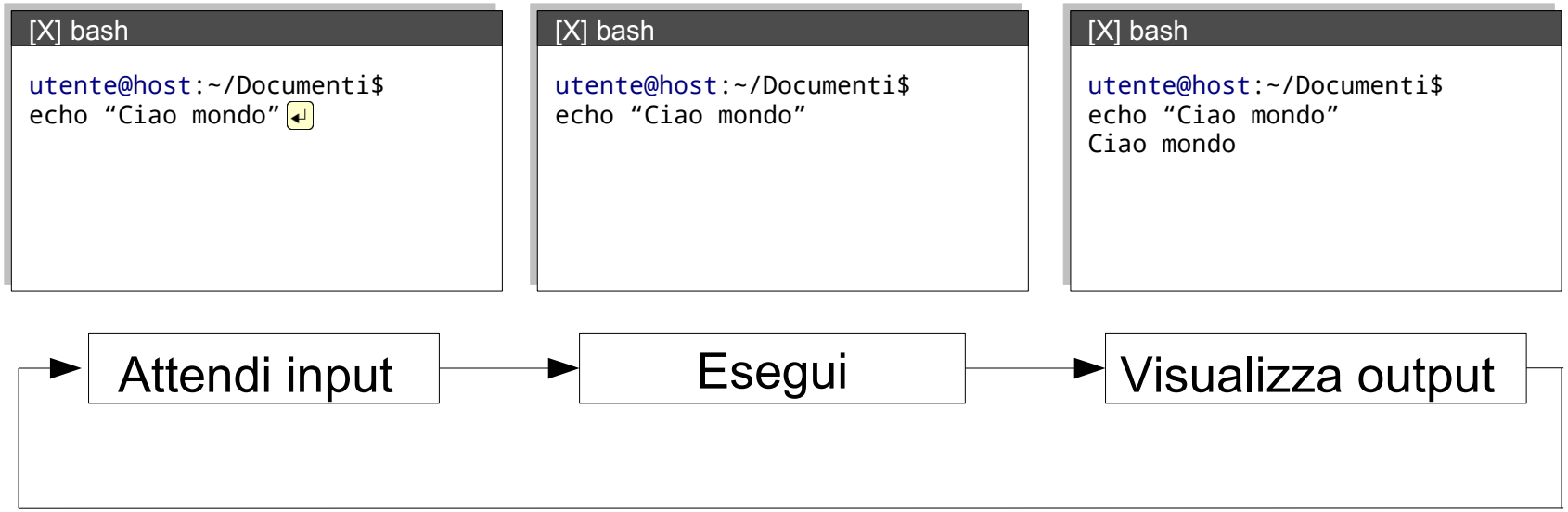
Utenti e gruppi

- Ogni utilizzatore è identificato da un nome utente
  - Il sistema identifica gli utenti con un identificatore numerico (UID)
  - Unix memorizza la lista degli utenti del sistema nel file `/etc/passwd`
- Gli utenti possono essere organizzati in gruppi
  - La lista dei gruppi è memorizzata in `/etc/groups`
  - Ogni gruppo ha un nome (es. Impiegati) e un identificatore numerico (GID)
  - Un utente puo` appartenere a più gruppi (es. Impiegati, Contabilità,...)
- È possibile assegnare i privilegi per l'accesso alle risorse del computer ai singoli utenti o a gruppi
  - es. permettere l'accesso al file Finanze.txt solo agli utenti del gruppo Contabilità
- Un utente “amministratore” (nome utente: root) ha tutti i privilegi sul sistema

Il prompt dei comandi

- `utente@host:~/Documenti$`



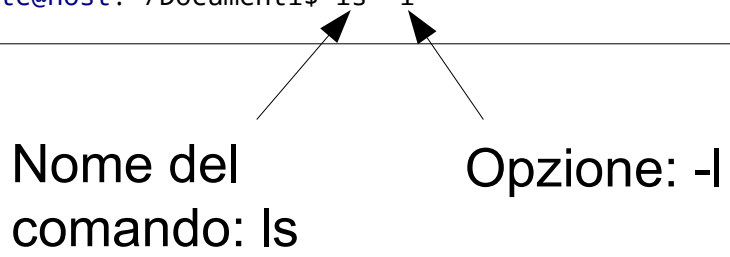


I comandi

- I comandi sono caratterizzati da
  - Nome del comando (ev. con il suo percorso completo)
  - Opzioni o flag (-)
    - Molte opzioni hanno una versione estesa (--opzione) e una compatta (-o)
  - Parametri

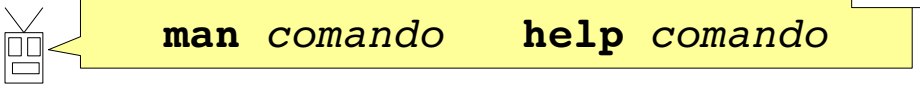
[X] bash

```
utente@host:~/Documenti$ ls -l
```



  - Un comando può essere “interno” (built-in, implementato dal programma della shell) o “esterno” se è definito in un programma separato

Aiuto ai comandi



Come si usa un determinato comando?

[X] bash

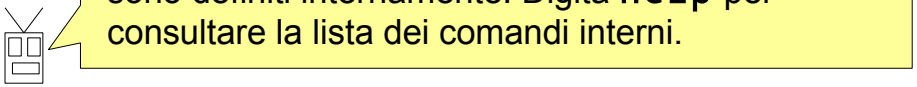
```
utente@host:~/Documenti$ man ls
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor –sort.
...
```

*Qual'è la differenza tra **man** e **help**?*



Aiuto ai comandi: man

man *sezione termine*

- Il manuale è organizzato in sezioni e sottosezioni, ognuna con diverse pagine
  - 1 Commands
  - 2 System Calls
  - 3 Library Functions
  - 4 Administrative Files
  - 5 Miscellaneous Information
  - 6 Games
  - 7 I/O and Special Files
  - 8 Maintenance Commands
- Se la sezione non è specificata, visualizza la prima occorrenza del termine
- In ogni sezione c'è una pagina **`intro`** che ne illustra il contenuto

[X] bash

```
utente@host:~/Documenti/Privato$ man 6 intro
INTRO(6)                   Linux Programmer's Manual                  INTRO(6)

NAME
       intro - Introduction to games
```

Struttura di una pagina di man

[X] bash

```
CAT(1)                  User Commands                  CAT(1)

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...

DESCRIPTION
       Concatenate FILE(s), or standard input, to standard output.

       -A, --show-all
              equivalent to -vET

       -b, --number-nonblank
              number nonempty output lines

       --help display this help and exit

       --version
              output version information and exit

       With no FILE, or when FILE is -, read standard input.

EXAMPLES

       cat f - g
              Output f's contents, then standard input, then g's
              contents.
```

Sintassi del comando [] per parametri opzionali

Descrizione dei parametri

q per uscire

Aiuto ai comandi: whatis, apropos

whatis *termine*

- Mostra solo la sezione NAME della pagina del manuale (equivale a **`man -f`**)

[X] bash

```
utente@host:~/Documenti/Privato$ whatis cat
cat (1)              - concatenate files and print on the standard output
```

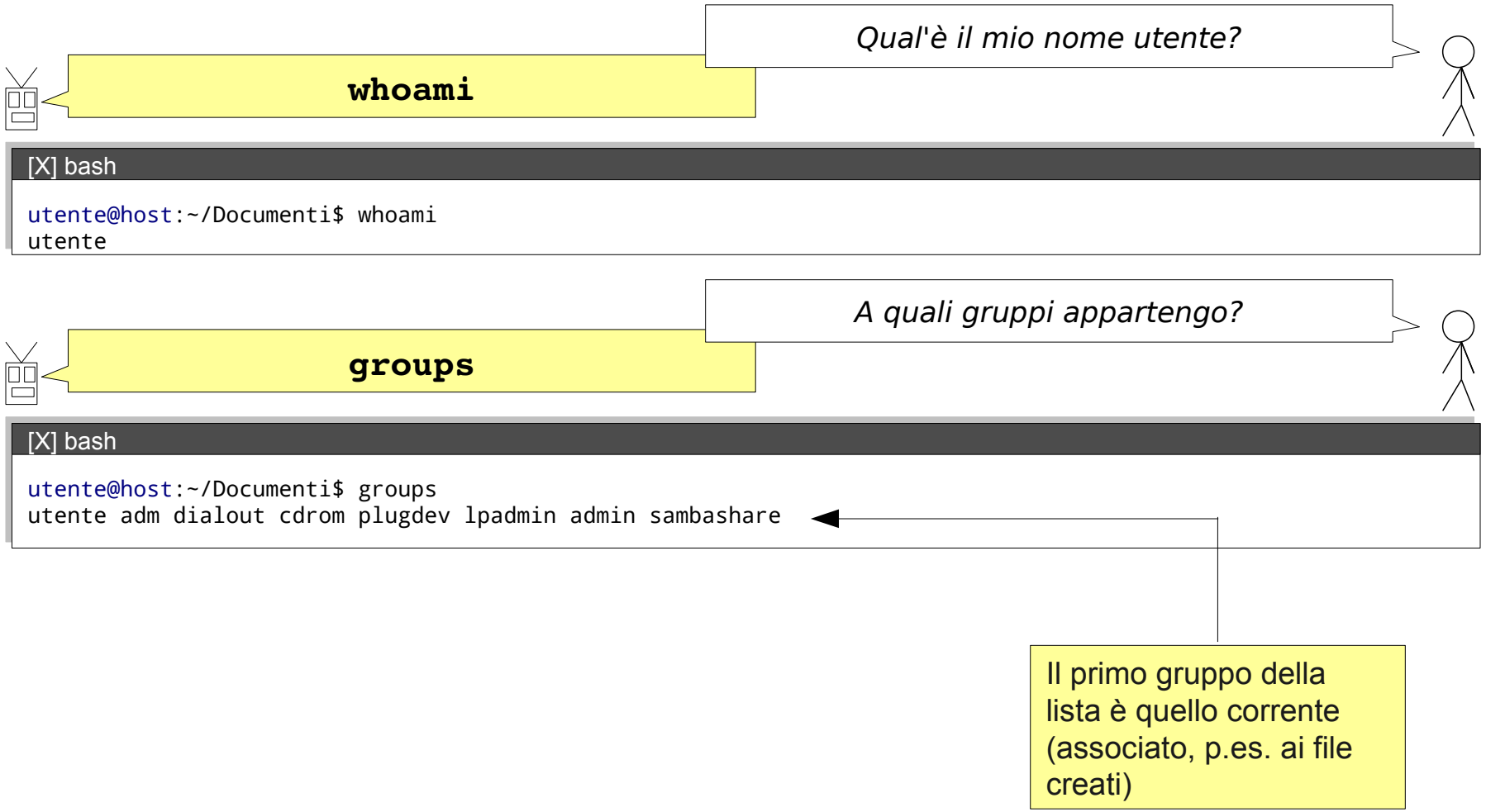
apropos *termine*

- Cerca i comandi la cui descrizione nel manuale (NAME) contiene le parole specificate (equivale a **`man -k`**)
  - Il comando non distingue fra maiuscole e minuscole

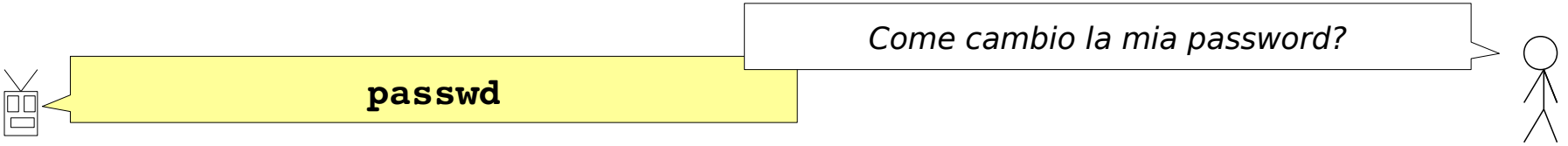
[X] bash

```
utente@host:~/Documenti/Privato$ apropos cat
catman (8)           - crea o aggiorna le pagine di manuale preformattate
PAM (7)              - Pluggable Authentication Modules for Linux
```

Utenti e gruppi

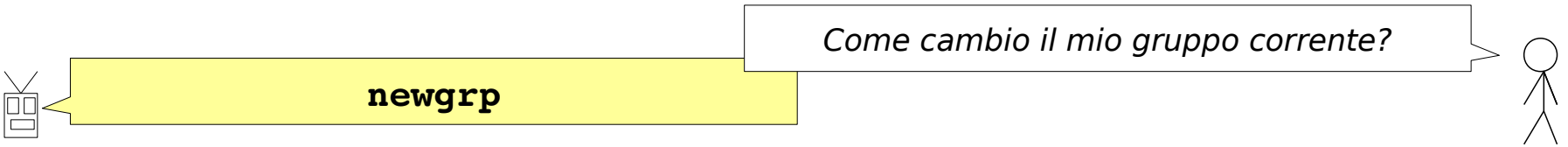


Utenti e gruppi



[X] bash

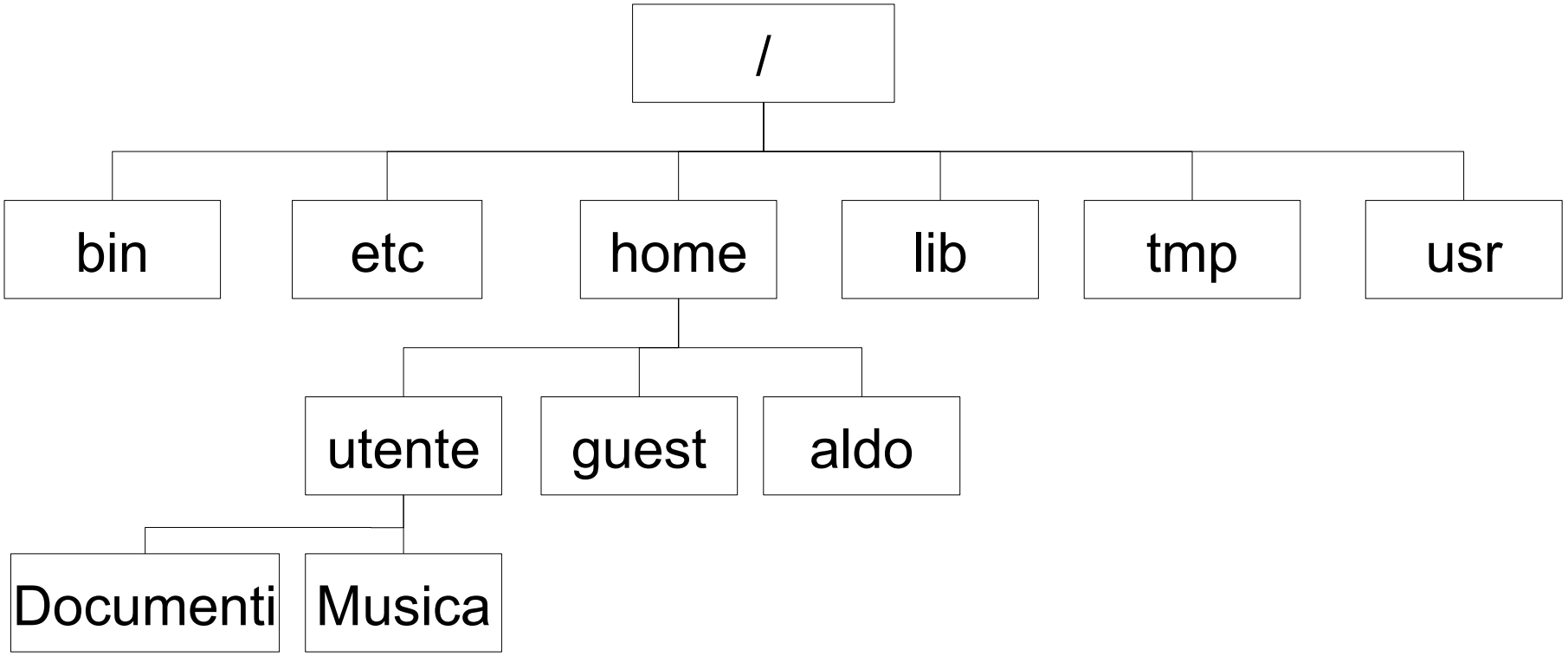
```
utente@host:~/Documenti$ passwd
Cambio password per utente.
Password UNIX (attuale):
Inserire nuova password UNIX:
Reinserire la nuova password UNIX:
```



[X] bash

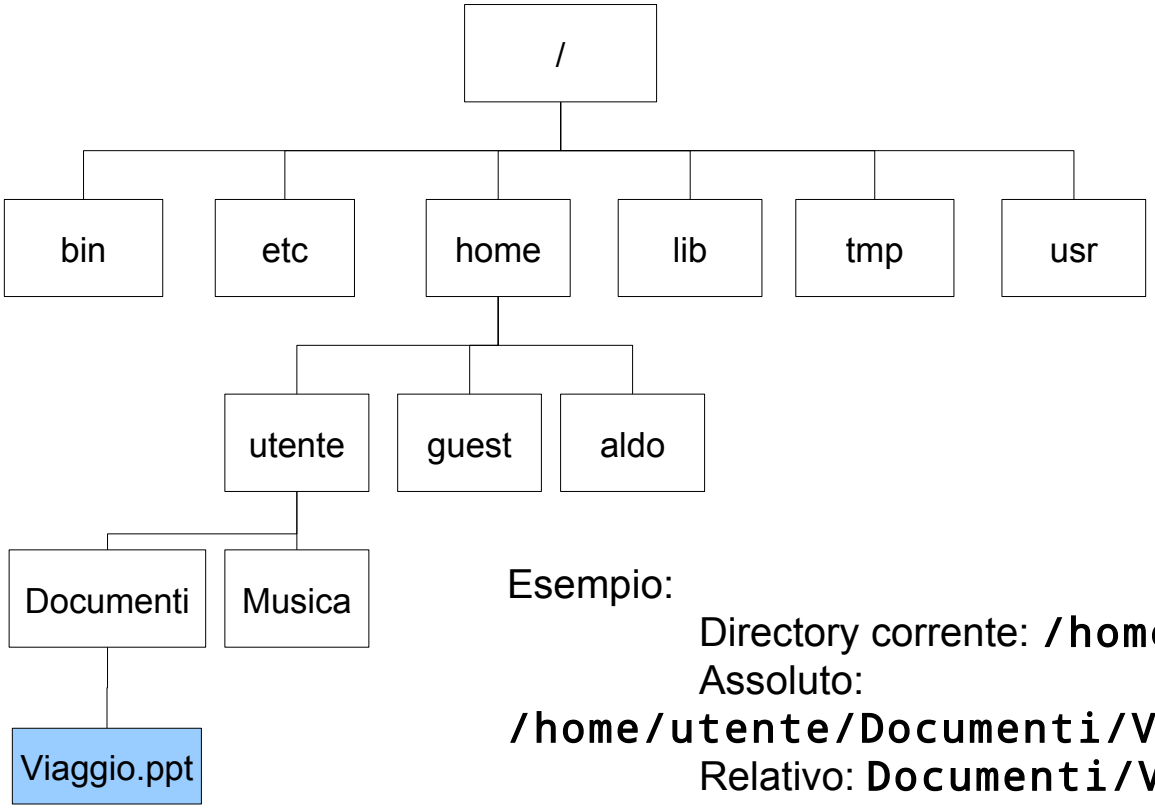
```
utente@host:~/Documenti$ groups
utente adm dialout cdrom plugdev lpadmin admin sambashare
utente@host:~/Documenti$ newgrp adm
utente@host:~/Documenti$ groups
adm utentedialout cdrom plugdev lpadmin admin sambashare
```

Filesystem Unix



Percorso assoluto e relativo

- È possibile specificare un file/directory attraverso il suo
  - Percorso assoluto (dalla radice /)
  - Percorso relativo (alla directory corrente)



Esempio:

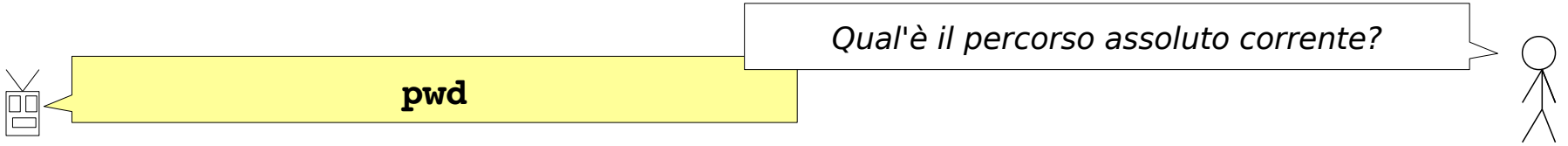
Directory corrente: **`/home/utente`**

Assoluto:

`/home/utente/Documenti/Viaggio.ppt`

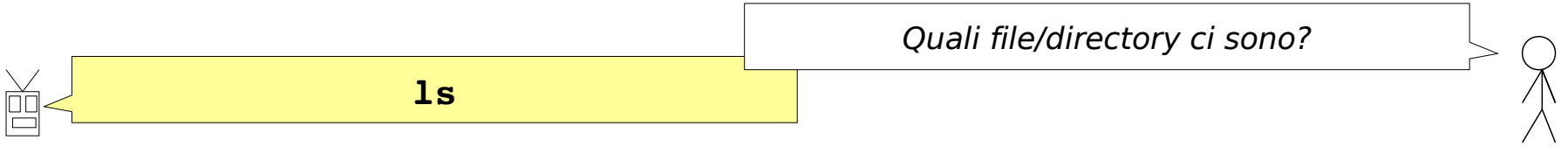
Relativo: **`Documenti/Viaggio.ppt`**

Navigare nel file system



[X] bash

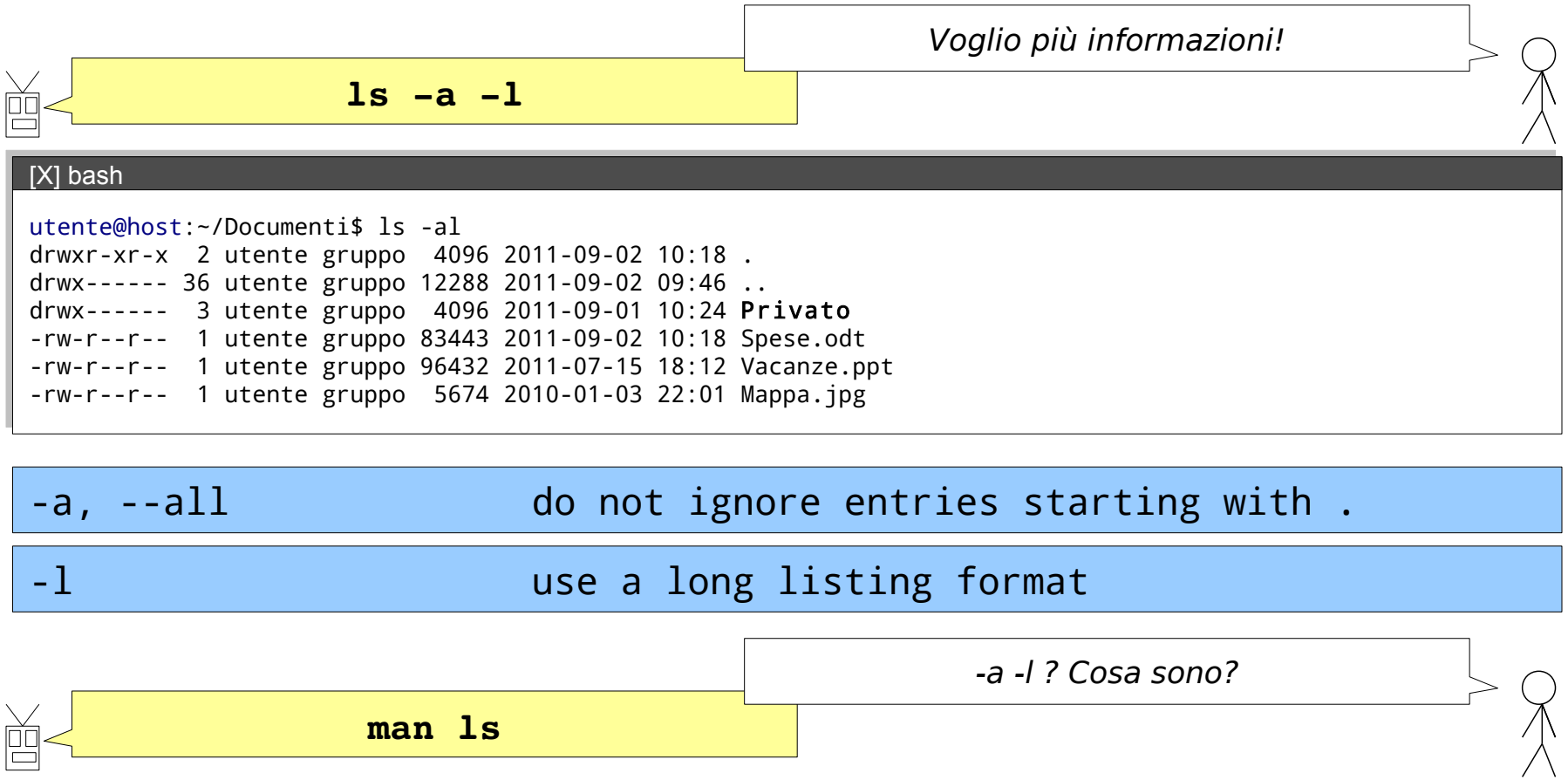
```
utente@host:~/Documenti$ pwd
/home/utente/Documenti
utente@host:~/Documenti$
```



[X] bash

```
utente@host:~/Documenti$ ls
Privato
Spese.odt
Vacanze.ppt
Mappa.jpg
utente@host:~/Documenti$
```

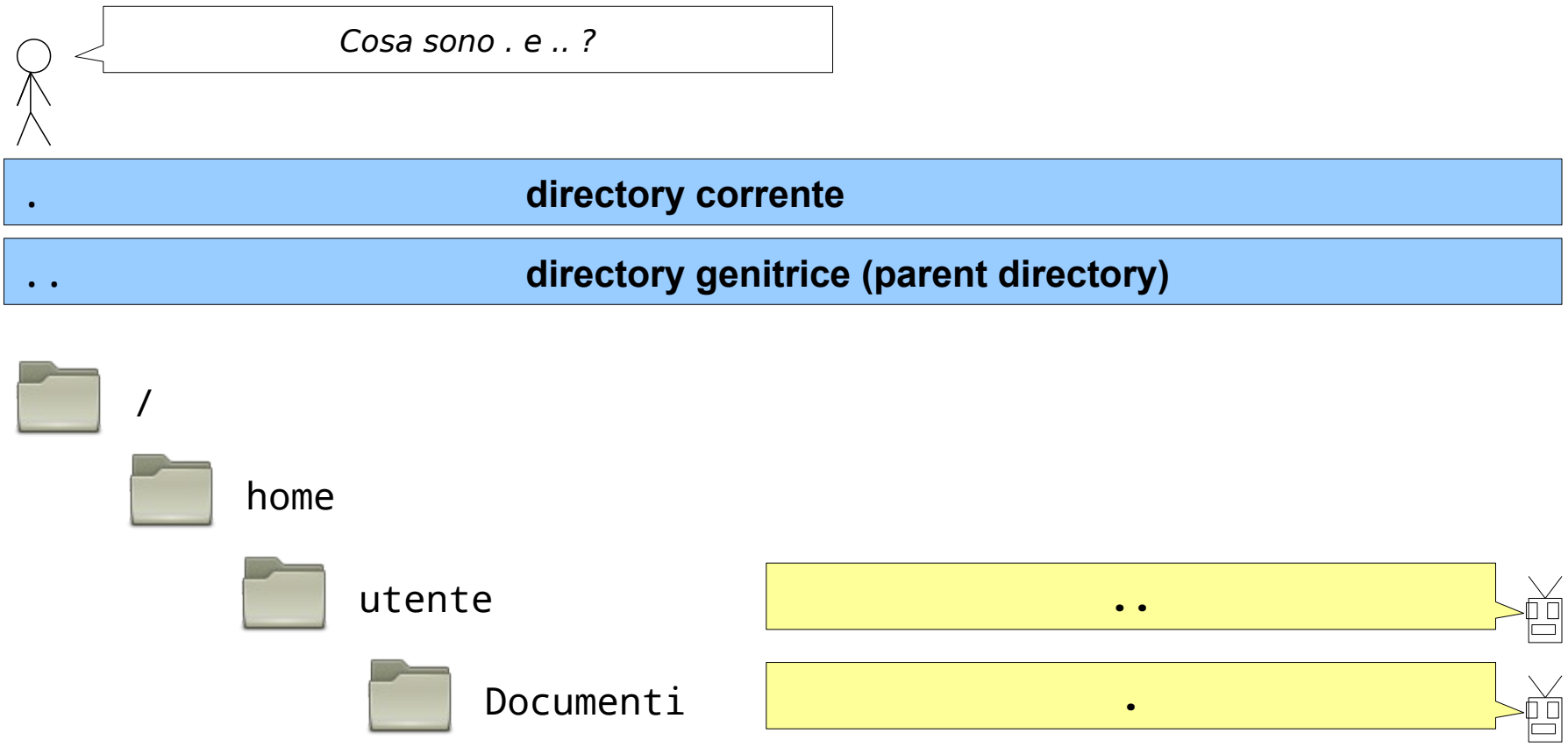
Navigare nel file system



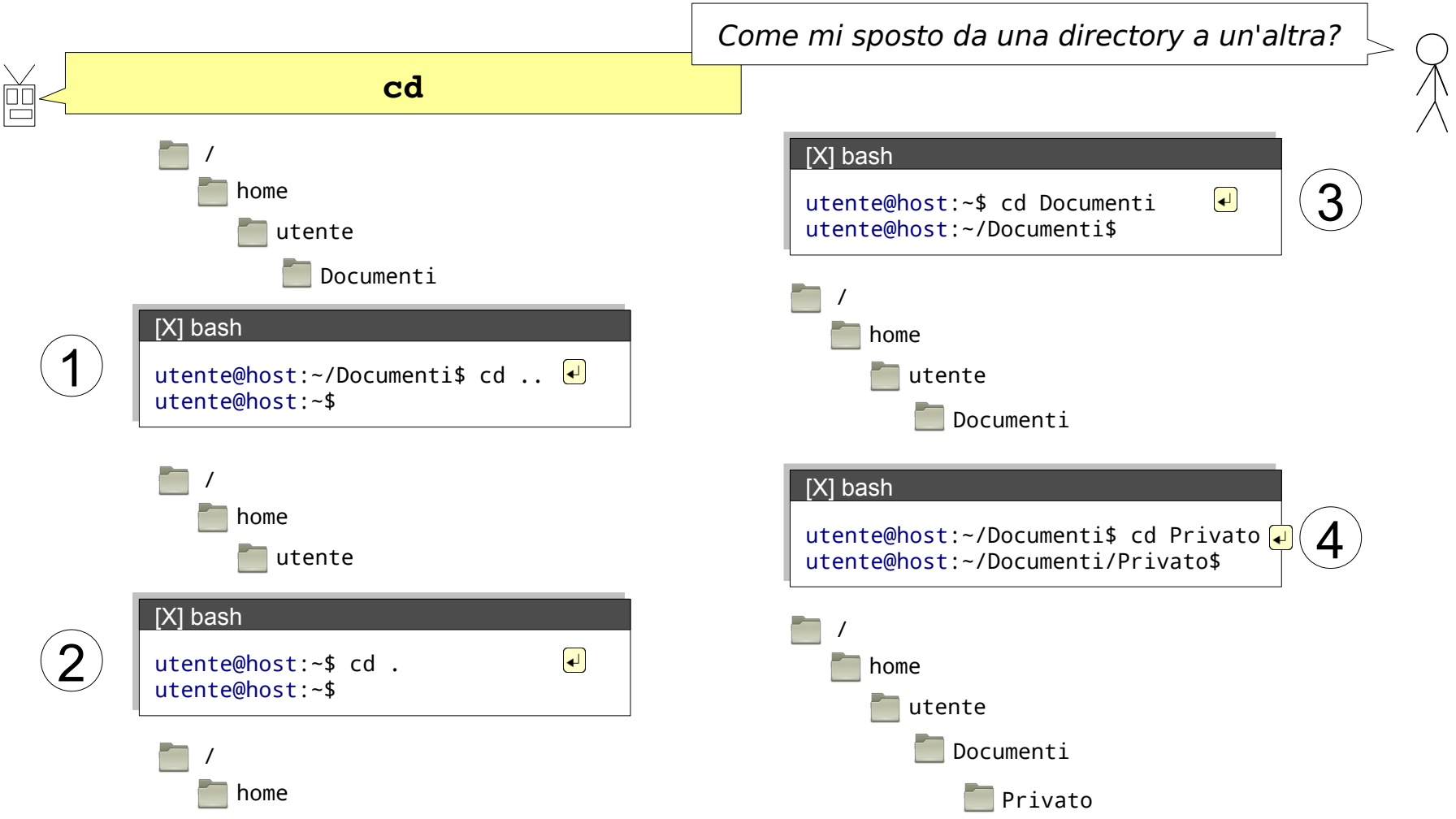
File nascosti

- I file/directory il cui nome inizia con un punto '.' non sono tipicamente visualizzati da ls (a meno di utilizzare l'opzione -a)
- Nella home directory sono presenti numerosi file nascosti, che contengono tipicamente le configurazioni dei programmi utente

Navigare nel file system



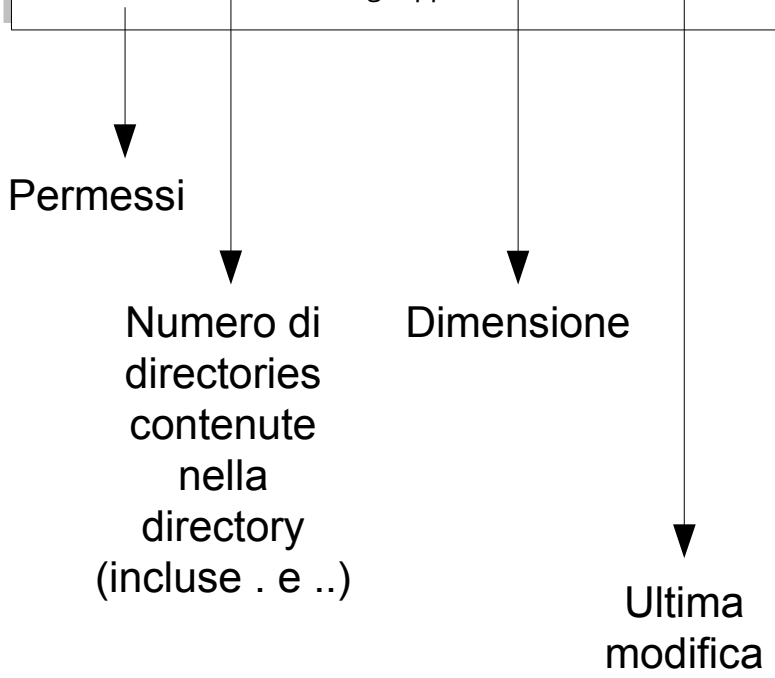
Navigare nel file system



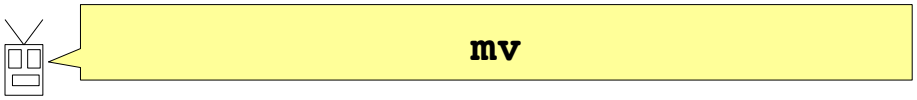
Proprietà di un file

[X] bash

```
utente@host:~/Documenti$ ls -l
drwx------  3 utente gruppo  4096 2011-09-01 10:24 Privato
-rw-r--r--  1 utente gruppo 83443 2011-09-02 10:18 Spese.odt
-rw-r--r--  1 utente gruppo 96432 2011-07-15 18:12 Vacanze.ppt
```



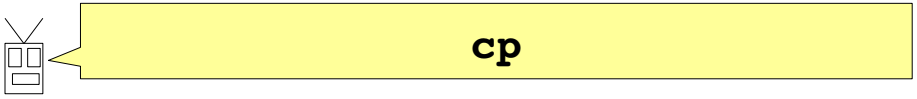
Rinominare, copiare, spostare un file



Rinominare = spostare nella stessa directory, utilizzando un nome diverso

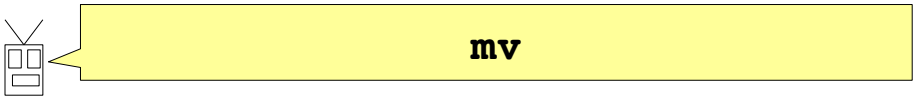
[X] bash

```
utente@host:~/Documenti$ mv Vacanze.ppt Montagna.ppt
```



[X] bash

```
utente@host:~/Documenti$ cp Spese.odt CopiaSpese.odt
utente@host:~/Documenti$ cp -r Privato BackupPrivato
```



[X] bash

```
utente@host:~/Documenti$ mv Copiaspese.odt Privato/
```

Creare file e directory

- Creare una directory

mkdir

[X] bash

```
utente@host:~/Documenti$ mkdir Foto
```

- Creare un file vuoto

touch

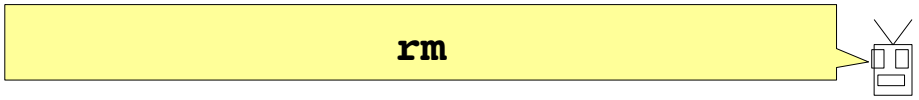
[X] bash

```
utente@host:~/Documenti$ touch Bottiglia
```

- Il vero scopo di touch è quello di aggiornare l'ora e la data di modifica di un file all'ora e data correnti. Se il file non esiste ne viene creato uno vuoto.

Eliminare file e directory

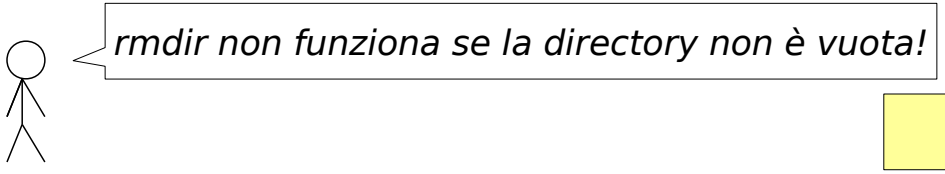
- Cancellare un file

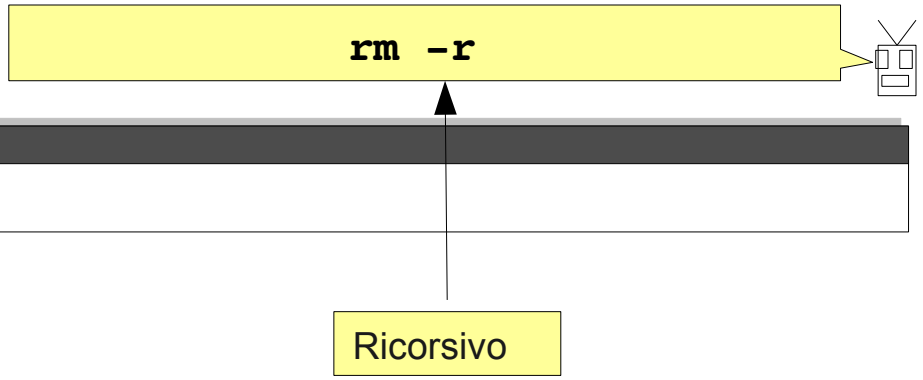



```
[X] bash
utente@host:~/Documenti$ rm Montagna.jpg
```

- Cancellare una directory


```
[X] bash
utente@host:~/Documenti$ rmdir Foto
```






```
[X] bash
utente@host:~/Documenti$ rm -r Foto
```

Autocompletamento e altre funzioni utili

[X] bash

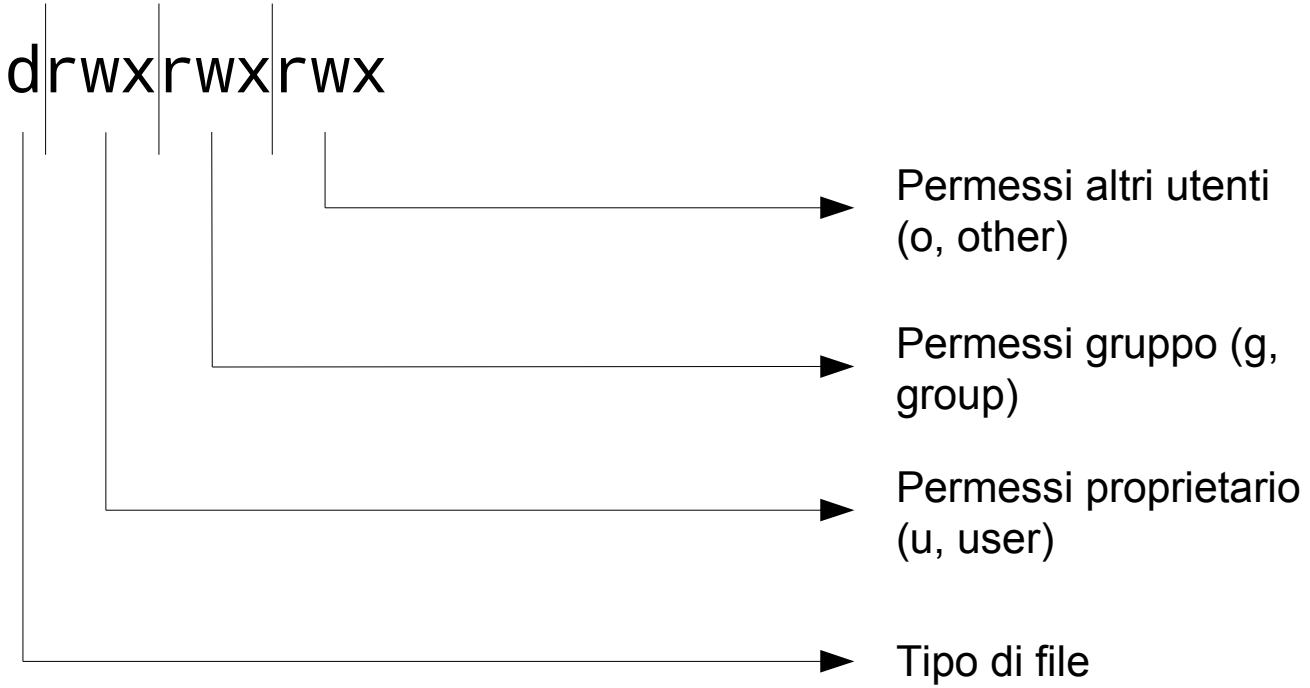
```
utente@host:~/Documenti$ pwd
/home/utente/Documenti
utente@host:~/Documenti$ cd Pri        TAB
utente@host:~/Documenti$ cd Privato
```

CTRL P	**Va al comando precedente**
CTRL N	**Va al comando successivo**
CTRL R	**Ricerca tra i comandi precedentemente utilizzati**
`history`	**Visualizza lo storico dei comandi utilizzati**
`!!`	**Riprende l'ultimo comando utilizzato**
`!n`	**Riprende l'n-esimo comando nello storico**
`!-n`	**Riprende l'n-ultimo comando nello storico**
`!bla`	**Riprende più recente che inizia con bla**

Altri comandi

`ls`	Lista dei files
`cd`	Cambia directory
`pwd`	Directory corrente
`cp`	Copia file
`mv`	Sposta/Rinomina file
`echo`	Scrivi una stringa sullo schermo
`date`	Data corrente
`seq`	Genera sequenza di numeri
`find`	Trova file
`grep`	Trova all'interno di file di testo
`cat`	Visualizza contenuto file
`wc`	Conta parole e linee
`less`	Visualizza il contenuto di un file, pagina per pagina
`cat`	Visualizza il contenuto di un file

Permessi


drwxrwxrwx
Permessi altri utenti (o, other)
Permessi gruppo (g, group)
Permessi proprietario (u, user)
Tipo di file

Tipi di file

-	**file regolare**
d	**directory**
b	**dispositivo a blocchi (es. harddisk)**
c	**dispositivo a caratteri (es. porta seriale, mouse,...)**
l	**collegamento simbolico (soft-link)**
p	**named pipe**
s	**Unix socket**

Permessi base

- **lettura** (r)
  - File: permette di leggerne il contenuto
  - Directory: consente di elencare contenuto (file, sotto-directory)
- **scrittura** (w)
  - File: permette di modificarne il contenuto
  - Directory: permette di aggiungere o rimuovere contenuto
    - **Posso cancellare file se ho permessi su directory che lo contiene, non mi serve il permesso di scrittura sul file specifico!**
- **esecuzione** (x)
  - File: permette di eseguirli
  - Directory: permette di attraversarle per accedere ai file ed alle sottodirectory
    - **Non permette di elencare contenuto!**

Permessi avanzati

- **set user ID (setuid o suid)**
  - Il file viene eseguito con i privilegi del proprietario del file e non dell'utente che esegue il file
  - Il permesso di esecuzione del campo proprietario può quindi assumere i valori:
    - **x** : permesso di esecuzione, setuid disabilitato
    - **S** : nessun permesso di esecuzione, setuid abilitato
    - **s** : permesso di esecuzione, setuid abilitato

Permessi avanzati

- **set group ID (setgid)**
  - Se applicato a un file:
    - Il file viene eseguito con i privilegi del gruppo del proprietario del file e non con quelli del gruppo dell'utente che esegue il file
  - Se applicato a una directory
    - I nuovi file e sottodirectory creati all'interno sono assegnati al gruppo della directory anziché al gruppo dell'utente che crea il file o la directory
  - Il permesso di esecuzione del campo gruppo può quindi assumere i valori:
    - **x** : permesso di esecuzione, setgid disabilitato
    - **S** : nessun permesso di esecuzione, setgid abilitato
    - **s** : permesso di esecuzione, setgid abilitato

Permessi avanzati

- **sticky**
  - Se applicato a un file: mantieni codice in memoria
  - Se applicato ad una directory:
    - i file contenuti possono essere cancellati e spostati solamente da:
      - utenti che ne sono proprietari
      - utente proprietario della directory
      - amministratore
  - Il permesso di esecuzione del campo altri utenti può quindi assumere i valori:
    - **x** : permesso di esecuzione, sticky disabilitato
    - **T** : nessun permesso di esecuzione, sticky abilitato
    - **t** : permesso di esecuzione, sticky abilitato

Cambiare i permessi (chmod)

chmod *permessi file*

- Modalità ottale

[X] bash

```
utente@host:~/Documenti$ chmod 666 Miofile
```

- Modalità simbolica

[X] bash

```
utente@host:~/Documenti$ chmod ugo=rw Miofile
```

  - u=user, g=group, o=others
- Modalità simbolica relativa
  - Il cambiamento puo' essere relativo alle permissions esistenti usando + o - invece di =

[X] bash

```
utente@host:~/Documenti$ chmod g+x Miofile
```

- Solo il proprietario può cambiare i permessi di un file!
  - Eccezione: l'amministratore di sistema (root)

Modalità ottale

- Da una a quattro cifre; le cifre sulla sinistra possono essere omesse (vengono sostituite con 0)
- In ogni posizione i privilegi sono determinati dalla somma dei valori ottali:

Prima cifra: set user ID (4) set group ID (2) sticky (1)	**Utente** **Seconda cifra:** leggi (4) scrivi (2) esegui (1)	**Gruppo** **Terza cifra:** leggi (4) scrivi (2) esegui (1)	**Altri** **Quarta cifra:** leggi (4) scrivi (2) esegui (1)

- Esempi:
  - Utente: R W X, Gruppo: R W, Altri: nessun permesso **0760** (oppure **760**)
  - Utente: R W X, Gruppo: R, Altri: R **0744** (oppure **744**)
  - Utente: R W X, Gruppo: R X, Altri: R X + Sticky **1755**

Cambiare il proprietario e gruppo

chown chgrp

- Cambiare proprietario

[X] bash

```
utente@host:~/Documenti$ chown mario Miofile
```

- Cambiare gruppo

[X] bash

```
utente@host:~/Documenti$ chgrp nuovogruppo Miofile
```

- È possibile utilizzare il parametro -r per applicare i cambiamenti ricorsivamente a tutti i file e sotto-directory

Permessi predefiniti

umask

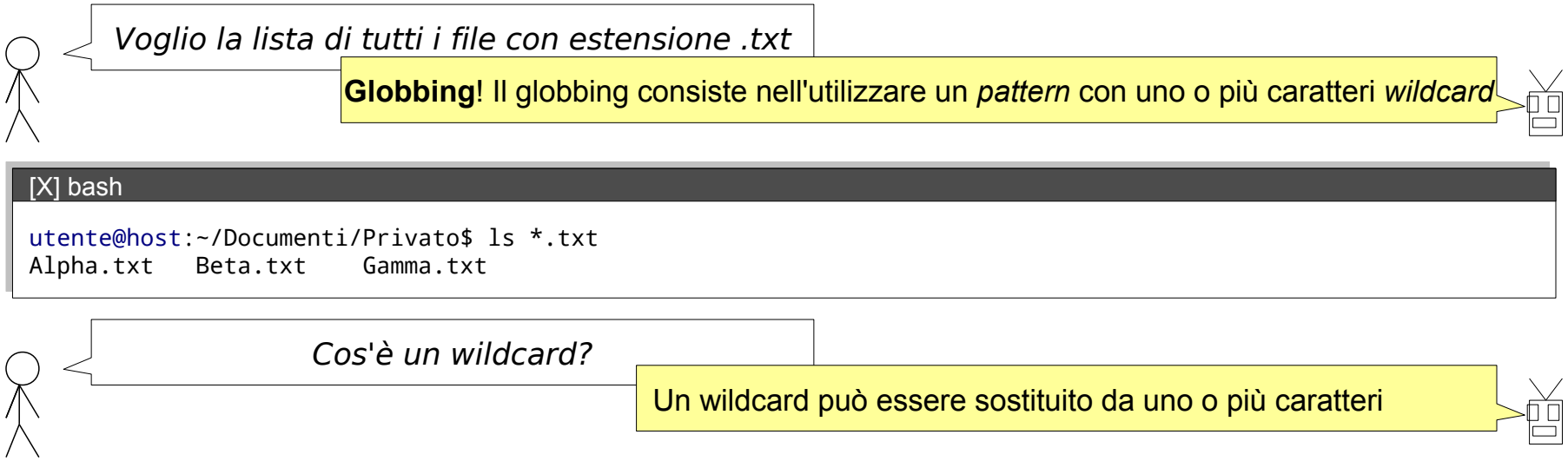
- Alla creazione di un file, Unix assegna i seguenti permessi:
  - Per i files ordinari non eseguibili: rw-rw-rw (666)
  - Per i files ordinari eseguibili e per directories: rwx rwx rwx (777)
- Il comando umask sottrae il numero ottale che si passa come parametro a quelle di default:

[X] bash

```
utente@host:~/Documenti$ umask 0022
```

Nuovo default per file: 644

Globbing (wildcard, carattere jolly, metacaratteri)



`*`	**corrispondenza con uno o più caratteri qualsiasi**
`?`	**corrispondenza con esattamente un carattere**
`[]`	**corrispondenza tra un gruppo di caratteri**
`[a-z]`	**un carattere dalla 'a' alla 'z'**
`[^abc]`	**non-corrispondenza tra un gruppo di caratteri**

Classi di caratteri

`[[:alnum:]]`	numeri o lettere (equivalente a [A-Za-z0-9])
`[[:alpha:]]`	lettere (equivalente a [A-Za-z])
`[[:blank:]]`	spazi o tabulazioni
`[[:cntrl:]]`	caratteri di controllo
`[[:digit:]]`	numeri (equivalente a [0-9])
`[[:graph:]]`	caratteri grafici (ASCII 33 - 126)
`[[:lower:]]`	lettere minuscole (equivalente a [a-z])
`[[:upper:]]`	lettere maiuscole (equivalente a [A-Z])
`[[:print:]]`	caratteri grafici più lo spazio (ASCII 32 - 126)
`[[:space:]]`	spazio, tabulatore, ritorno a capo,...
`[[:xdigit:]]`	numero esadecimale (equivalente a [0-9A-Fa-f])

Globbing

Elenca tutti i file il cui nome termina con una lettera

[X] bash

```
utente@host:~/Documenti/Privato$ ls *[a-zA-Z]
```

- Pattern:

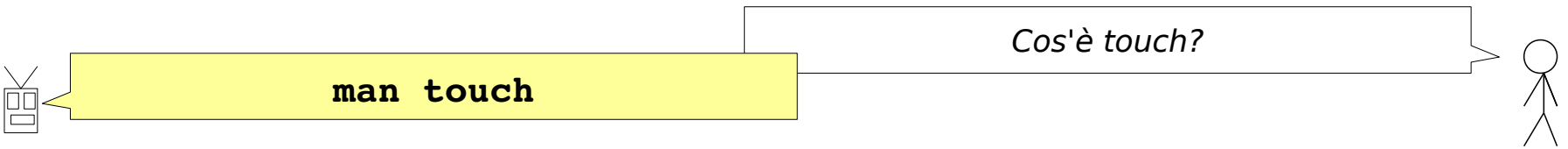
Qualsiasi carattere, anche ripetuto → `*[a-zA-Z]` ← "Un carattere dalla 'a' alla 'z' o dalla 'A' alla 'Z'

- Il pattern matching viene effettuato prima di eseguire il comando
  - se il matching è possibile (c'è almeno un file che corrisponde), la lista dei nomi di file corrispondenti sostituisce il pattern
  - Altrimenti, il pattern non viene sostituito

Globbing

[X] bash

```
utente@host:~/Documenti$ touch Timmy.pig Tommy.pig Jimmy.pig
```



[X] bash

```
utente@host:~/Documenti$ ls *.pig
Timmy.pig Tommy.pig Jimmy.pig
utente@host:~/Documenti$ ls T*.pig
Timmy.pig Tommy.pig
utente@host:~/Documenti$ echo “I tre porcellini“ *.pig
I tre porcellini Jimmy.pig Timmy.pig Tommy.pig
utente@host:~/Documenti$ rm *.pig
utente@host:~/Documenti$ ls *.pig
ls: impossibile accedere a *.pig: File o directory non esistente
utente@host:~/Documenti$ echo "I tre porcellini" *.pig
I tre porcellini *.pig
```

Il globbing viene espanso prima di eseguire il comando:
`ls T*.pig` diventa **`ls Timmy.pig Tommy.pig`**

Alternanza

*Voglio la lista di tutti i file con estensione .jpg o *.JPG o *.Jpg o *.jPG o ...*

Alternanza! Permette di generare un insieme di combinazioni

[X] bash

```
utente@host:~/Documenti/Privato$ ls *.{j,J}{p,P}{g,G}
```

{} **genera un insieme di combinazioni**

Ma se una combinazione non corrisponde a un file ottengo un errore!

[X] bash

```
utente@host:~/Documenti/Privato$ ls *.{j,J}{p,P}{g,G}
ls: impossibile accedere a *.jpG: File o directory non esistente
ls: impossibile accedere a *.jPg: File o directory non esistente
ls: impossibile accedere a *.jPG: File o directory non esistente
ls: impossibile accedere a *.Jpg: File o directory non esistente
ls: impossibile accedere a *.JpG: File o directory non esistente
ls: impossibile accedere a *.JPg: File o directory non esistente
Alpha.jpg  Beta.JPG
```

shopt -s nullglob

Variabili

- È possibile definire delle variabili associate a dei valori
  - Una variabile è accessibile solo all'interno della stessa shell
- Prima dell'esecuzione di un comando avviene la sostituzione degli eventuali riferimenti a variabili con i valori delle variabili stesse.

[X] bash

```
utente@host:~/Documenti$ messaggio="Ciao mondo"
utente@host:~/Documenti$ echo $messaggio
Ciao mondo
```

`nome=valore`	**Definisce una nuova variabile**
`$nome`	**Accesso a una variabile**

Tilde

- Il simbolo tilde **~** viene espanso nel percorso della directory personale (Home) dell'utente corrente
  - Viene espanso solo se si trova all'inizio di una parola o se è separato

[X] bash

```
utente@host:~/Documenti$ cd ~
utente@host:~$ cd ~/Documenti
utente@host:~/Documenti$
```

- **~+** è il percorso della directory corrente (equivale a pwd)
- **~-** è il percorso della directory corrente precedente (prima dell'ultimo cd).
  - Se non si è mai cambiata la directory nella shell corrente, ~- non viene espanso!

[X] bash

```
utente@host:~/Documenti$ echo ~+
/home/utente/Documenti
utente@host:~/Documenti$ cd ~
utente@host:~$ echo ~-
/home/utente/Documenti
```

Command substitution

- C’é la possibilità di includere un comando in un altro e fare in modo che il comando incluso venga eseguito prima e sostituito sulla linea di comando principale con il suo output.
- Il comando incluso viene delimitato con gli apici “ ` ” oppure con `$(...)`

[X] bash

```
utente@host:~/Documenti$ echo `pwd`
/home/utente/Documenti
utente@host:~/Documenti$ echo $(date)
gio 8 set 2011, 16.18.53, CEST
```

Alias

- Un alias permette di assegnare un nuovo nome (in maniera più generale, una qualsiasi sequenza di caratteri) a un comando

```
alias nome='sostituzione'
```

[X] bash

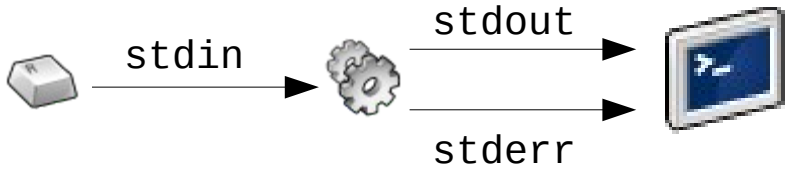
```
utente@host:~/Documenti$ alias listafile='ls -al'
utente@host:~/Documenti$ listafile
drwxr-xr-x  2 utente gruppo  4096 2011-09-02 10:18 .
drwx------ 36 utente gruppo 12288 2011-09-02 09:46 ..
drwx------  3 utente gruppo  4096 2011-09-01 10:24 Privato
-rw-r--r--  1 utente gruppo 83443 2011-09-02 10:18 Spese.odt
-rw-r--r--  1 utente gruppo 96432 2011-07-15 18:12 Vacanze.ppt
-rw-r--r--  1 utente gruppo  5674 2010-01-03 22:01 Mappa.jpg
```

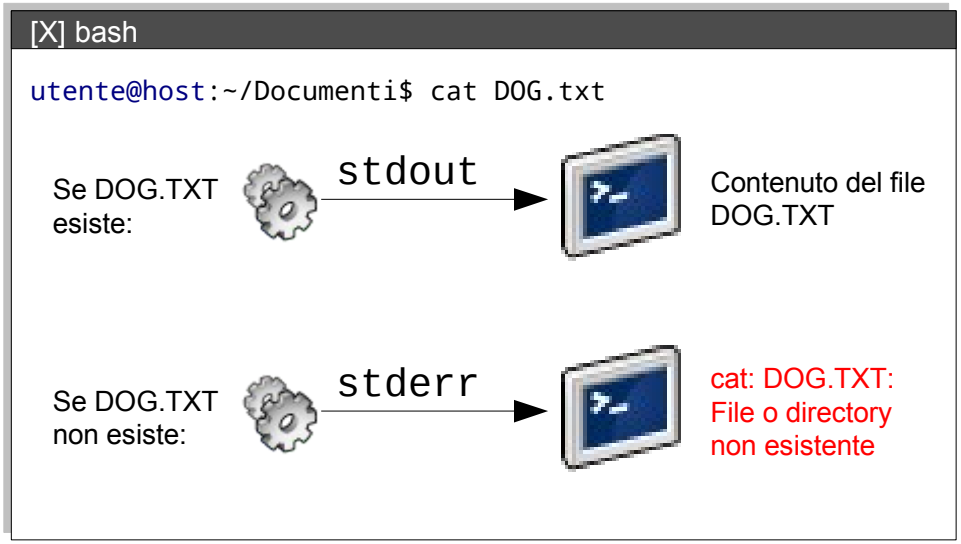
stdin, stdout, stderr, redirezione

Sui sistemi Unix i programmi hanno accesso a tre *stream* di input e output:

- **stdin** (standard input) [0]
  - Input dalla tastiera
- **stdout** (standard output) [1]
  - Output su schermo
- **stderr** (standard error) [2]
  - Output degli errori su schermo

- Ogni stream è associato a un numero detto file descriptor (fd)





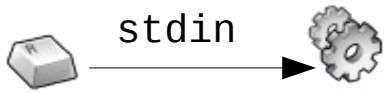
stdin, stdout, stderr, redirezione

[X] bash

Lettura da stdin

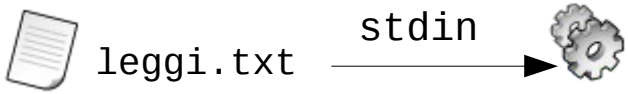
```
utente@host:~/Documenti/Privato$ cat
ciao
ciao
sole
sole
```

CTRL D



Redirezione di stdin da un file

```
utente@host:~/Documenti/Privato$ cat < leggi.txt
```



Redirezione di stdout su un file (sovrascrivi)

```
utente@host:~/Documenti/Privato$ cat leggi.txt > scrivi.txt
```

Redirezione di stdout su un file (aggiunta)

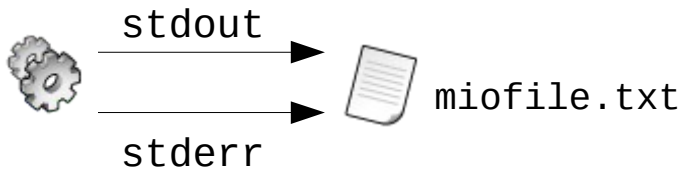
```
utente@host:~/Documenti/Privato$ cat leggi.txt >> aggiungi.txt
```

stdin, stdout, stderr, redirezione

[X] bash

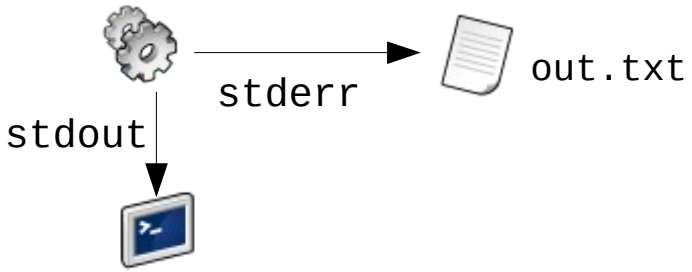
Redirezione di stdout e stderr

```
utente@host:~/Documenti/Privato$ ls *.bogus >& miofile.txt
```



Redirezione di stderr

```
utente@host:~/Documenti/Privato$ cat leggi.txt 2> out.txt
```



Redirezione di stdout a stderr

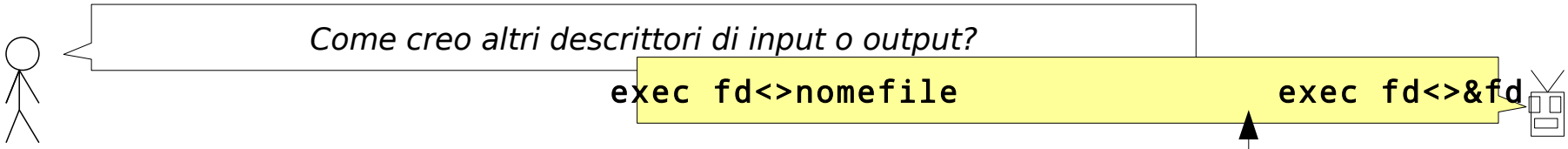
```
utente@host:~/Documenti/Privato$ cat leggi.txt 1>&2
```

Redirezione di stderr a stdout

```
utente@host:~/Documenti/Privato$ cat leggi.txt 2>&1
```

Se viene omesso & l'output di stdout viene scritto in un file chiamato '2'

stdin, stdout, stderr, redirezione



[X] bash

```
utente@host:~/Documenti$ exec 3<>Datalog.txt
utente@host:~/Documenti$ ls -l /proc/$$/fd
lr-x------. 1 utente utente 64  9 set 16:23 0 -> /dev/pts/1
lrwx------. 1 utente utente 64  9 set 16:23 1 -> /dev/pts/1
lrwx------. 1 utente utente 64  9 set 16:23 2 -> /dev/pts/1
lrwx------. 1 utente utente 64 12 set 10:49 255 -> /dev/pts/1
lrwx------. 1 utente utente 64  9 set 16:23 3 -> /home/utente/Datalog.txt
```

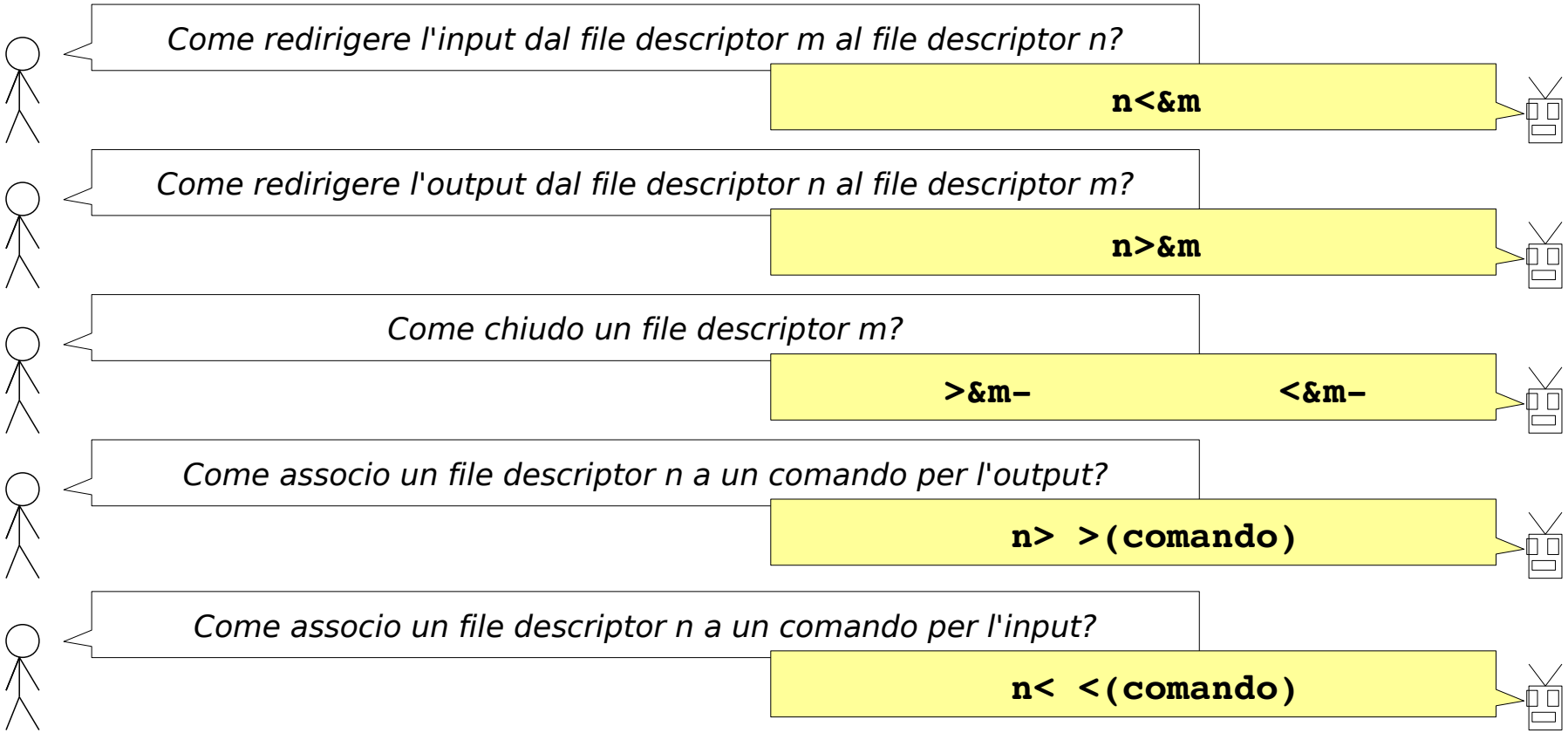
< sola lettura

> sola scrittura

<> lettura e scrittura

Crea una nuova shell e esegue il comando

stdin, stdout, stderr, redirezione



Esempi di redirezione

[X] bash

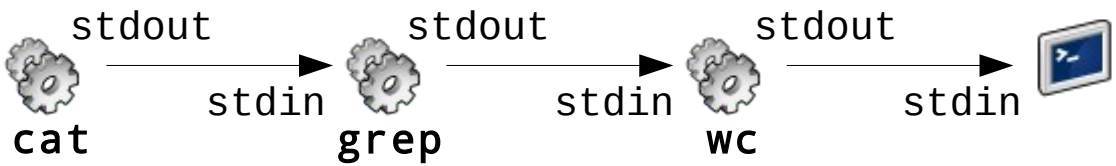
```
utente@host:~/Documenti$ exec 55< <(echo "ciao mondo")
utente@host:~/Documenti$ cat <&55
ciao mondo
utente@host:~/Documenti$ exec 55<&-
utente@host:~/Documenti$ echo "Ambienti operativi" > Corso.txt
utente@host:~/Documenti$ exec 99<>Corso.txt
utente@host:~/Documenti$ ls -l /proc/$$/fd
totale 0
lr-x------. 1 utente utente 64 12 set 11:01 0 -> /dev/pts/3
lrwx------. 1 utente utente 64 12 set 11:01 1 -> /dev/pts/3
lrwx------. 1 utente utente 64 12 set 11:01 2 -> /dev/pts/3
lrwx------. 1 utente utente 64 12 set 11:01 255 -> /dev/pts/3
lrwx------. 1 utente utente 64 12 set 11:36 99 -> /home/utente/Corso.txt
utente@host:~/Documenti$ echo "Redirezione in bash" >&99
utente@host:~/Documenti$ exec 99>&-
utente@host:~/Documenti$ cat Corso.txt
Redirezione in bash
utente@host:~/Documenti$
```

Pipe

- Una pipe è un buffer che permette di “collegare” tra loro più programmi e redirigere stdout verso stdin
  - Per creare una pipe si utilizza il carattere | Alt Gr 1 *
- Opera in modalità FIFO
  - First-in, First-out: “Il primo carattere ad entrare è il primo ad uscire”

[X] bash

```
utente@host:~/Documenti$ cat dati.txt | grep “Ciao” | wc -l
```



* Su una tastiera con layout svizzero francese

Filtrare l'output di un comando

head

Voglio le prime 50 righe del file DivinaCommedia.txt

[X] bash

```
utente@host:~/Documenti/Privato$ head -n 50 DivinaCommedia.txt
utente@host:~/Documenti/Privato$ cat DivinaCommedia.txt | head -n 50
```

Voglio tutto il contenuto di Testo.txt tranne le prime 3 righe

[X] bash

```
utente@host:~/Documenti/Privato$ cat Testo.txt | head -n-3
```

tail

Voglio le ultime 50 righe del file DivinaCommedia.txt

[X] bash

```
utente@host:~/Documenti/Privato$ tail -n 50 DivinaCommedia.txt
utente@host:~/Documenti/Privato$ cat DivinaCommedia.txt | tail -n 50
```

Filtrare l'output di un comando

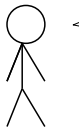
uniq

Visualizza il contenuto di Dati.txt omettendo le righe adiacenti uguali

[X] bash

```
utente@host:~/Documenti/Privato$ cat Dati.txt | uniq
```

tr



Voglio sostituire tutte le 'a' con delle 'b'

[X] bash

```
utente@host:~/Documenti/Privato$ tr a b
Ciao
Cibo
```

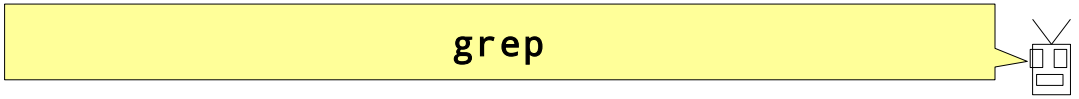
cut

Voglio estrarre i primi 5 caratteri di ogni linea

[X] bash

```
utente@host:~/Documenti/Privato$ cut -c 1-5 miofile
```

Filtrare l'output di un comando



Visualizza i files nella directory corrente che contengono la stringa "ciao"

[X] bash

```
utente@host:~/Documenti/Privato$ ls | grep ciao
```

Visualizza le righe di Telefono.txt che contengono la parola Giovanni

[X] bash

```
utente@host:~/Documenti/Privato$ grep Giovanni Telefono.txt
```

Nota: le potenzialità di grep verranno esplorare nelle lezioni sulle espressioni regolari

Cercare un file: find

- Cercare nel filesystem i file con certe caratteristiche

find percorso *[espressione]*

- Discende ricorsivamente le directory specificate del percorso, cercando tutti i files che rendono vera l'espressione (formata da una o più opzioni)

[X] bash

```
utente@host:~/Documenti/Privato$ find . -name "*.odp"
```

Doppio apice per evitare globbing!

[X] bash

```
utente@host:~/Documenti/Privato$ find . -type f -name "*.odp"
```

[X] bash

```
utente@host:~/Documenti/Privato$ find . -type f -mtime -1 -name "*.odp"
```

[X] bash

```
utente@host:~/Documenti/Privato$ find . -type f -mtime -1 -name "*.odp" -exec echo Trovato {}  \;
```

Cercare un file: find

[X] bash

```
utente@host:~/Documenti/Privato$ find . -name "*.odp" -or -name "*.ppt"
```

[X] bash

```
utente@host:~/Documenti/Privato$ find . -name "*.odp" -size +200k
```

[X] bash

```
utente@host:~/Documenti/Privato$ find . -name "*.odp" -not -user andrea
```

[X] bash

```
utente@host:~/Documenti/Privato$ find . -name "*.odp" -perm -o=rw
```

[X] bash

```
utente@host:~/Documenti/Privato$ find . ! -name "*.odp" -perm -o=rw
```

[X] bash

```
utente@host:~/Documenti/Privato$ find . -name "*.txt" ! -wholename "*/backup/*" -exec cp {} backup/ \;
```

Ordinare l'output di un comando: sort

- Permette di ordinare l'output di un comando o visualizzare il contenuto ordinato di un file

[X] bash

```
utente@host:~$ cat Nomi.txt
Giovanni
Aldo
Giacomo
due
4
Due
2
1
3
uno
tre
utente@host:~$ sort Nomi.txt
1
2
3
4
Aldo
due
Due
Giacomo
Giovanni
tre
uno
```

Sort

[X] bash

```
utente@host:~$ cat Nomi.txt | sort
1
2
3
4
Aldo
due
Due
Giacomo
Giovanni
tre
Uno
utente@host:~$ cat Nomi.txt | sort -r
uno
tre
Giovanni
Giacomo
Due
due
Aldo
4
3
2
1
```

Con l'opzione -r inverte l'ordine

Quoting

- Le stringhe possono essere delimitate da tre tipi di apici/virgolette:
  - doppie “”
    - causano l'espansione delle variabili al loro interno
  - singole ''
    - nessuna espansione
  - Accenti gravi ``
    - eseguono la stringa e ne sostituiscono il valore con l'output del comando (equivalente a `$(...)`)

[X] bash

```
utente@host:~$ etichetta=”mondo”
utente@host:~$ echo “Ciao $etichetta”
Ciao mondo
utente@host:~$ echo 'Ciao $etichetta'
Ciao $etichetta
```

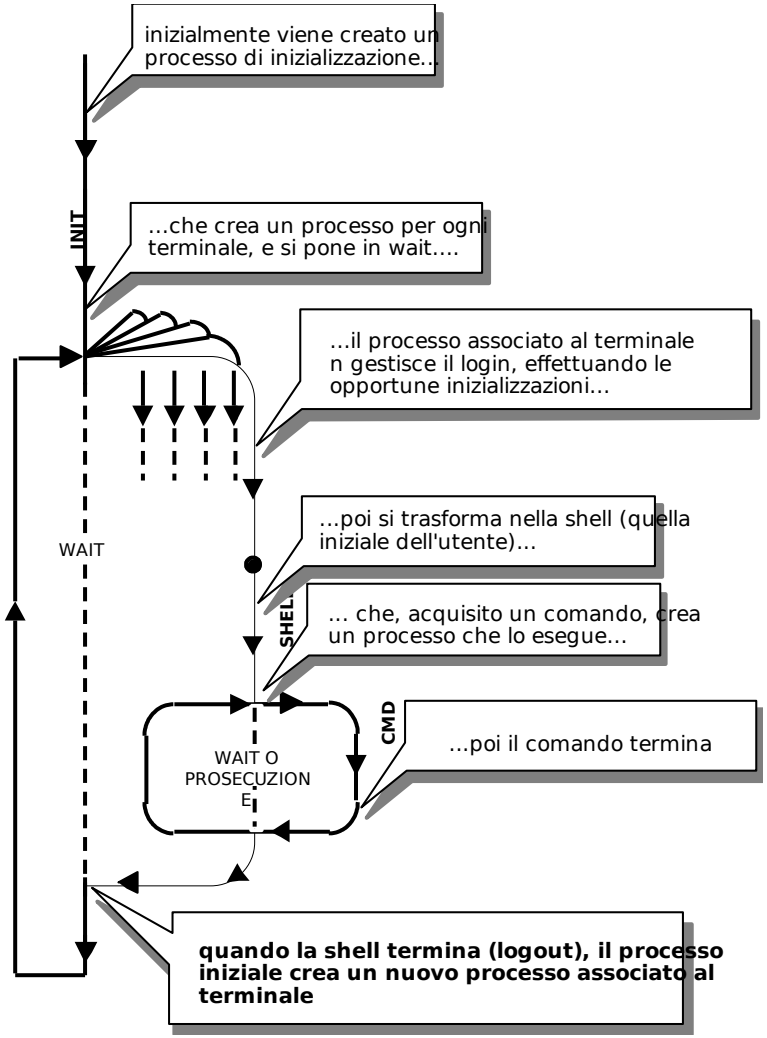
Escaping

- Possiamo impedire l'espansione delle variabili e il globbing
  - Il carattere \ (backslash) viene anteposto al carattere speciale $, ?, *, \, ...

[X] bash

```
utente@host:~$ etichetta=”mondo”
utente@host:~$ echo “Ciao \$etichetta”
Ciao $etichetta
utente@host:~$ echo *.txt
Esempio.txt Finanze.txt
utente@host:~$ echo \*.txt
*.txt
utente@host:~$ echo “Apici \””
Apici “
```

Gestione dei processi



Processi

- Ogni processo è identificato da un numero chiamato PID (Process Identifier)

- Con il comando `ps` posso vedere i processi in esecuzione
- I processi sono organizzati in maniera gerarchica
  - Quando eseguo un programma da shell, viene creato un sotto-processo
  - con il parametro `f` il comando ps visualizza “graficamente” la struttura gerarchica
  - con il parametro `e` posso vedere la lista di tutti i processi

[X] bash

```
utente@host:~$ ps f
  PID TTY      STAT   TIME COMMAND
26867 pts/3    Ss     0:00 bash
26947 pts/3    S      0:00  \_ bash
26960 pts/3    S      0:00      \_ bash
26972 pts/3    R+     0:00          \_ ps f
```

Sotto-processi

Variabili di ambiente

- Le variabili di ambiente possono modificare il comportamento dei programmi
  - I valori delle variabili di ambiente vengono “ereditati” dai sotto-processi
  - la variabile d'ambiente **PATH** indica il percorso predefinito per i programmi
    - Quando viene richiesto di eseguire un comando senza specificarne il percorso, il comando è ricercato nei percorsi definiti in PATH

[X] bash

```
utente@host:~/Documenti$ export pi=”Greco”
utente@host:~/Documenti$ bash
utente@host:~/Documenti$ echo $pi
Greco
```

`export nome=valore`	**Definisce una nuova variabile d'ambiente**
`$nome`	**Accesso a una variabile d'ambiente**

[X] bash

```
utente@host:~/Documenti$ export PATH=~/.local/bin:$PATH
utente@host:~/Documenti$ export PS1=blabla-
blabla-
```

Anche le variabili di ambiente vengono sostituite con il loro valore prima di eseguire il comando

Gestione dei lavori

- La gestione dei lavori con Bash permette di
  - Sospendere l'esecuzione di un comando

```
CTRL + Z
```

  - Gestire i comandi sospesi

```
jobs
```

  - Continuare l'esecuzione di un comando in background

```
bg
```

  - Continuare l'esecuzione di un comando in foreground

```
fg
```

  - Terminare l'esecuzione

```
CTRL + C
```

Esecuzione di un comando in background

- Un comando può essere eseguito in background (secondo piano)
  - L'esecuzione non blocca la shell corrente permettendo all'utente di continuare a inserire nuovi comandi

[X] bash

```
utente@host:~$ find . -name Importante > /tmp/Risultati.txt &
[1] 29010
utente@host:~$
```

Inviare dei segnali a un processo

- Posso inviare un segnale a un processo per sospenderlo, terminarlo, ecc.

```
kill -segnale PID
```

- Per fermare un processo in esecuzione si usa il segnale **9**

[X] bash

```
utente@host:~$ kill -9 29010
```

- Esistono anche altri segnali:
  - **TERM** **15** Terminazione del processo (è il segnale di default)
  - **INT** **2** Interruzione del processo (equivale a CTRL+C)
  - **KILL** **9** Terminazione del processo (non può essere ignorato nè trattato)
  - **STOP** **19** Sospensione del processo (non può essere ignorato nè trattato)
  - **CONT** **18** Ripartenza di un processo sospeso

- Con “man 7 signal” posso vedere tutti i segnali validi

Valore di ritorno / uscita

- Quando un comando termina la sua esecuzione viene ritornato un valore numerico di uscita / di stato o *exit status* (da 0 a 255)
  - Associato alla variabile **$?**
  - 0 (zero) indica che il comando è terminato correttamente (“nessun errore”)
  - Gli altri valori sono utilizzati per segnalare una condizione d'errore
    - I codici di errore sono solitamente indicati nel manuale del comando

[X] bash

```
utente@host:~$ true
utente@host:~$ echo $?
0
utente@host:~$ false
utente@host:~$ echo $?
1
utente@host:~$ date
utente@host:~$ echo $?
0
```