Editor di Testo

[...vi ...]

Salvatore Cuomo

Corso di Laurea in Informatica

UNIVERSITA' degli Studi di Napoli

FEDERICO II

Editor di Testo

- Vengono utilizzati per manipolare FILES nel File System di S.O. oppure per visualizzane il contenuto

- I piu' comuni editor di testo UNIX lavorano direttamente sul formato di testo ASCII e non necessitano di formati di file "speciali"

- I piu' comuni editor di testo sono: vi, emacs, pico

- Altri tipi di editor di testo basati su interfaccia grafica hanno bisogno di sistemi operativi ad interfaccia grafica (Window systems)

Utilizzeremo vi (Perche' ?)

- Disponibile su tutti i sistemi UNIX
- Puo' essere utilizzato in modalità remota (ovvero senza necessariamente utilizzare una interfaccia grafica per il sistema operativo)
- Ottimo bilanciamento tra semplicità di utilizzo e potenza.

Principali comandi: Modalità inserimento

Dalla modalità comando si può passare alla modalità inserimento in accordo con la tabella che segue.

i	prima del cursore
a	dopo il cursore
I	all'inizio della linea
A	alla fine della linea
o	apertura linea precedente
O	apertura linea successiva

N.B. In modalità inserimento tutti i caratteri digitati sulla tastiera vengono inseriti nel file

Principali comandi

Si accede a questa modalità mediante il tasto *Esc*

- **Spostamento del cursore**

l	una colonna a destra
k	una linea sopra
j	una linea sotto
h	una colonna a sinistra
$	fine linea
^	inizio linea
w	parola successiva
e	fine parola

N.B. Si resta comunque in modalità comando

- **Sostituzione**

cw	sostituzione della parola
cc	sostituzione della linea
C	fine di linea
r	carattere del cursore

Principali comandi (cont.)

- **Cancellazione**

dw	cancella la parola
dd	cancella la linea
D	alla fine di linea
x	carattere del cursore
X	carattere prima del cursore

- **Altre funzioni**

u	annulla
/	ricerca in avanti
?	ricerca a ritroso
n	instanza successiva
.	ripete l'ultimo comando
Y	aggiorna la linea
p	inserisce sotto
P	inserisce sopra
ZZ	scrive ed esce
Esc	cancella il comando

Modalità ultima linea

Dalla modalità comando si passa alla modalità ultima linea secondo la seguente tabella:

:w	scrive il file
:q	esce
:wq	scrive ed esce
:n	file successivo
:r	legge il file
:e	aggiorna il file
:f	nome del file
:set	aggiorna le opzioni
:!	Uscita dalla shell
:n	linea n

N.B. Alcune opzioni di set:

number (num) numera le linee del file

no auto indent (noai) evita il capoverso

Riepilogo:

3 modes of vi

command mode

you can navigate the file and use the commands shown on this page

insert mode

you can type into the file, and with vim you can still move around the file

last-line mode

you can issue complicated commands on the last line of the editor

editing

`i`	insert
`o`	open a new line (below)
`O`	open a new line (above)
`a`	append
`A`	append at end of line
`u`	undo
`.`	repeat last command

cutting and pasting

`yy`	yank (copy)
`5yy`	yank 5 lines
`dd`	delete current line
`6dd`	delete six lines
`p`	paste (below current line) (lower-case 'p')
`P`	paste (above current line) (capital 'P')

Riepilogo:

deleting

`x`	delete current character
`10x`	delete 10 characters
`dd`	delete current line
`6dd`	delete six lines
`d0`	delete to beginning of line
`d$`	delete to end of line

navigation – up, down

`[Up]`	move up one line
`5[Up]`	move up 5 lines
`[Down]`	move down 1 line
`6[Down]`	move down 6 lines
`1G`	go to line 1
`15G`	go to line 15
`G`	go to last line
`H`	go to top of screen (high)
`M`	go to middle of screen
`L`	go to bottom of screen (low)

navigation – left, right

`w`	go to next word
`7w`	move over 7 words
`b`	back one word
`0`	go to beginning of line
`$`	go to end of line

Riepilogo:

searching

`/foo`	search forward for "foo"
`?foo`	search backwards for "foo"
n	repeat last search

saving, exiting

`:w`	write contents to disk
`:wq`	write contents and quit
`ZZ`	write contents and quit
`:q`	quit
`:q!`	quit and don't save changes

miscellaneous

`:!ls`	run "ls" command from editor
`:r foo`	read file foo into this file
`:10,20d`	delete lines 10-20

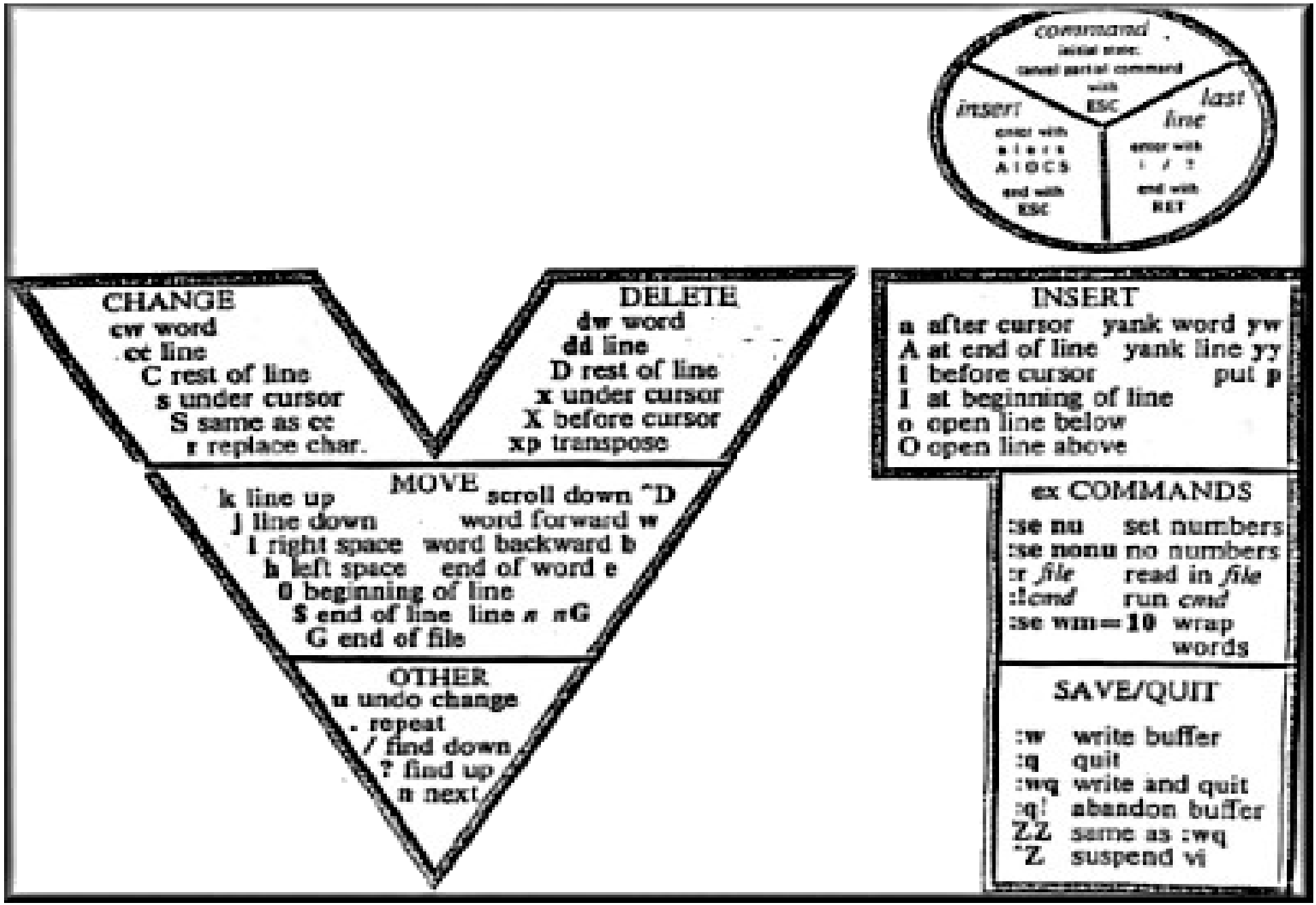
more miscellaneous

`:1,$s/foo/bar/g`

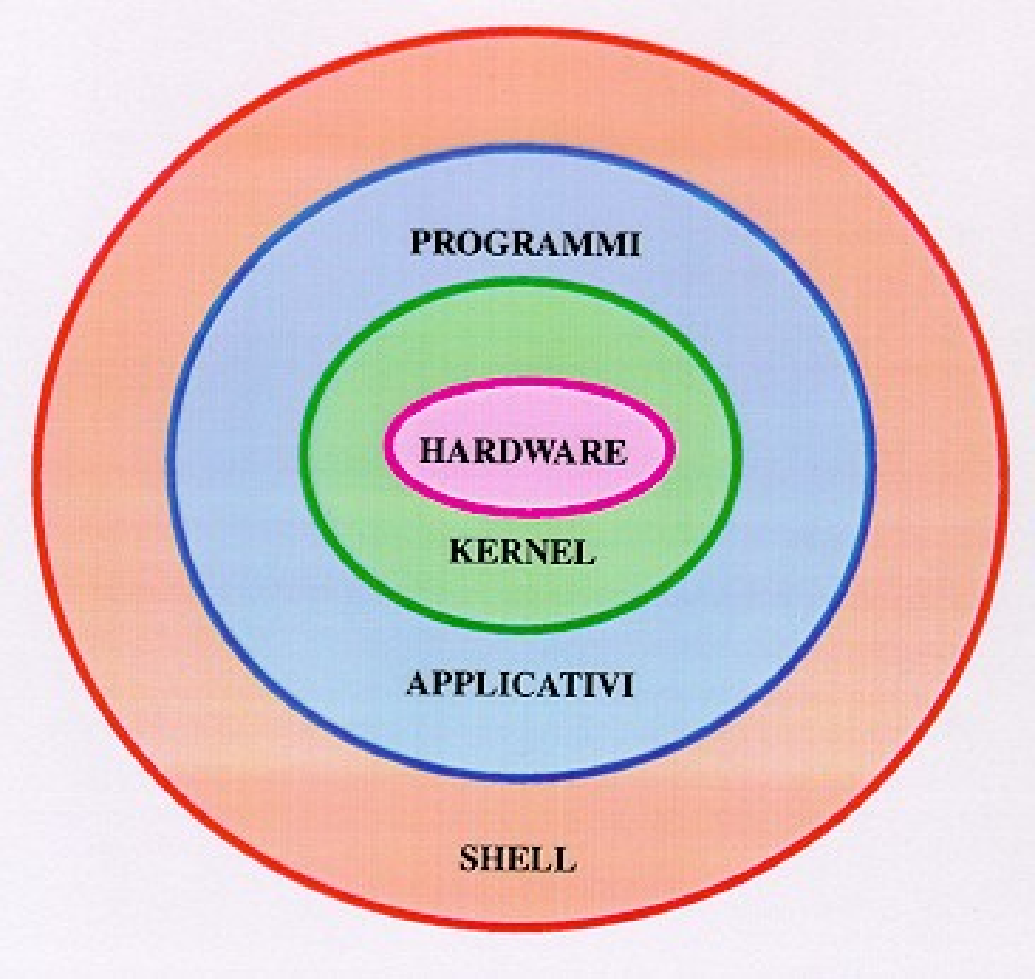
from the first line to the last line change all ocurrences of "foo" to "bar"

`first[Ctrl][p]`

vi auto-complete; turns `"first"` into `"firstName"`, assuming you have a variable in the file named `firstName`


command
ESC
insert
last line
CHANGE
cw word
cc line
C rest of line
s under cursor
S same as cc
r replace char.
DELETE
dw word
dd line
D rest of line
x under cursor
X before cursor
xp transpose
INSERT
a after cursor
A at end of line
i before cursor
I at beginning of line
o open line below
O open line above
yank word yw
yank line yy
put p
MOVE
k line up
j line down
l right space
h left space
0 beginning of line
$ end of line
G end of file
scroll down ^D
word forward w
word backward b
end of word e
line n nG
ex COMMANDS
:se nu set numbers
:se nonu no numbers
:r file read in file
:!cmd run cmd
:se wm=10 wrap words
OTHER
u undo change
. repeat
/ find down
? find up
n next
SAVE/QUIT
:w write buffer
:q quit
:wq write and quit
:q! abandon buffer
ZZ same as :wq
^Z suspend vi

La macchina virtuale di Unix

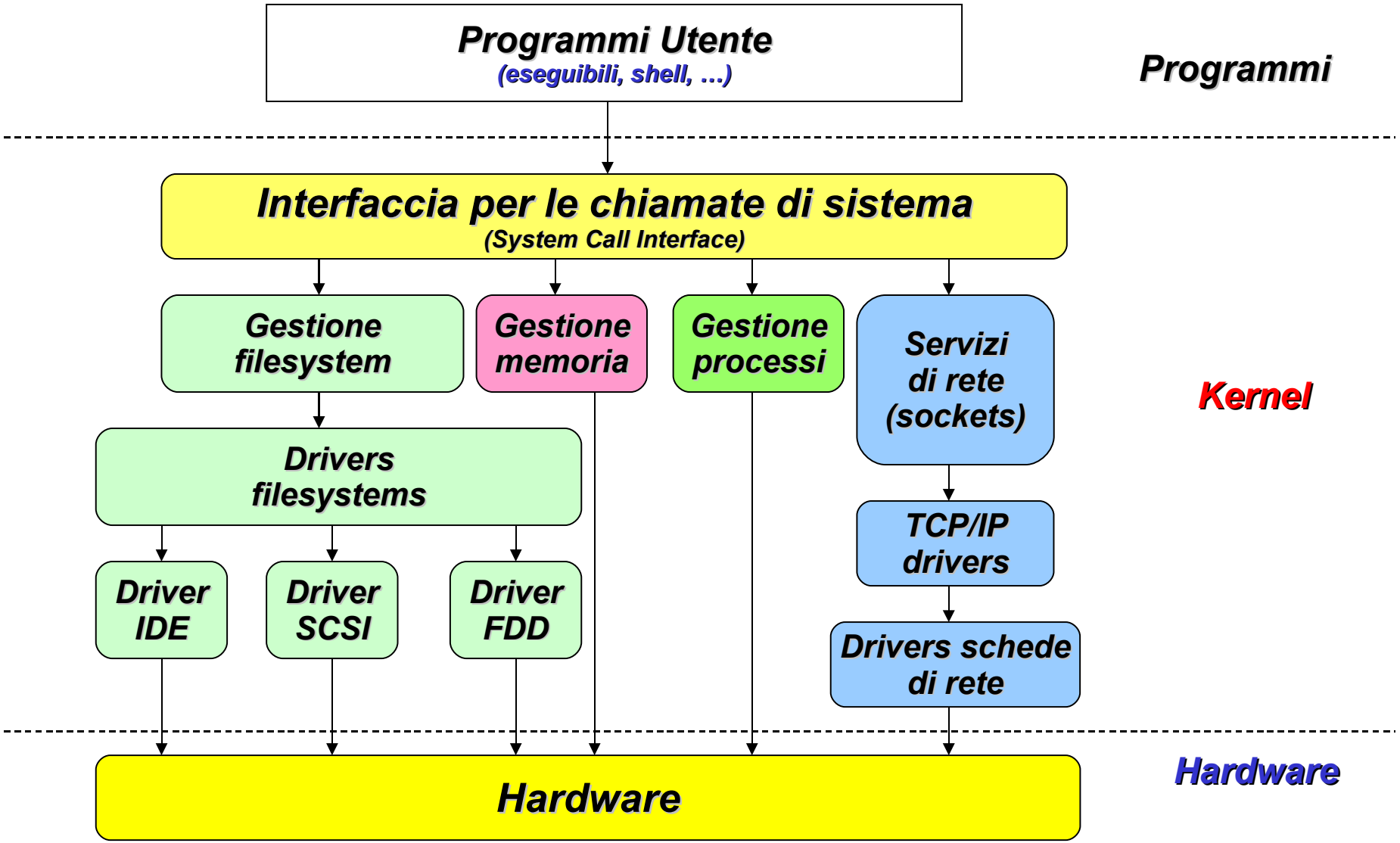


SHELL: Interprete del linguaggio di comando

PROGRAMMI APPLICATIVI: Risorse software del sistema operativo, programmi di sistema (editor, compilatori, …)

KERNEL: Nucleo del sistema operativo. Coordina l'esecuzione di tutti i programmi e l'accesso a tutte le risorse del sistema.

La macchina virtuale di Unix (cont.)



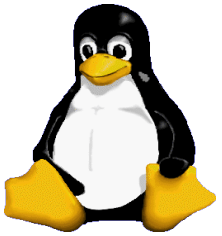
Sistemi operativi Unix-like (Unix Flavors)

Sistemi Operativi a pagamento

- AIX (IBM) AIX 5L (Linux)
- BeOS
- Digital Unix (Compaq)
- HP-UX (Hewlett Packard)
- IRIX (Silicon Graphics)
- MacOS X (Apple)
- Solaris (Sun Microsystems)

Sistemi Operativi non a pagamento

- Linux (Ubuntu, Fedora, Debian, Gentoo, SuSe,...)
- FreeBSD



Sistemi Operativi real-time

- LynxOS, QNX, ...

I sistemi operativi operano su un insieme di dati chiamati

FILE

Un file è una collezione di caratteri chiusa da un carattere speciale denominato *end-of-file*

Un file può essere organizzato in linee (record).

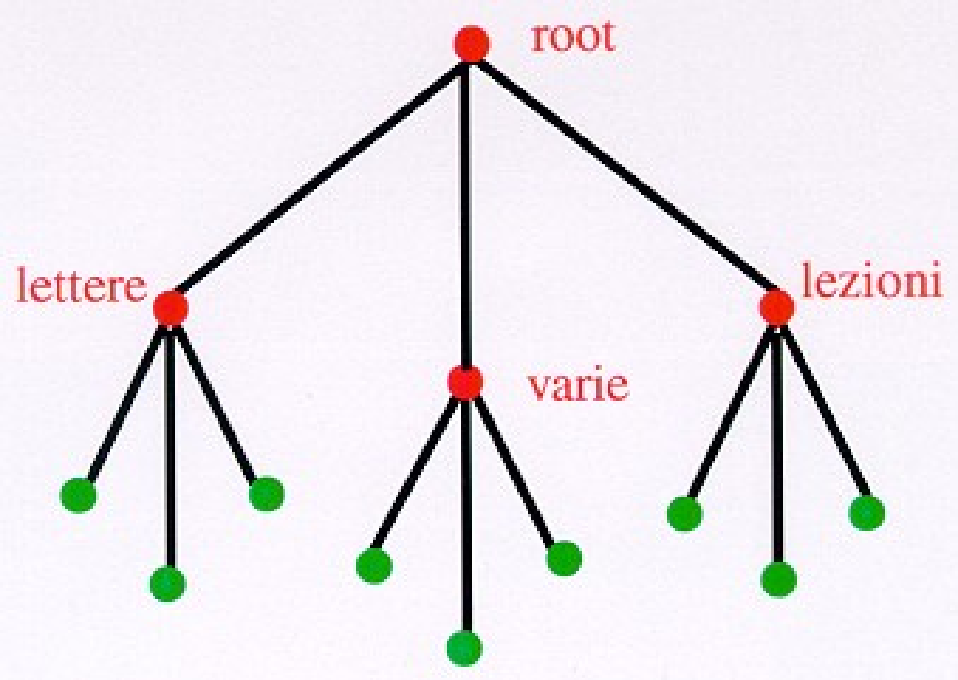
Può essere creato:

- mediante lettura da una unità di input
- come risultato dell'esecuzione di un programma.

Il file system

è il metodo logico di organizzazione dei file

Nei moderni sistemi operativi i file sono organizzati secondo una struttura gerarchica ad albero



- **file ordinari**
- **directory**

Il file system UNIX

È formato da tre tipi di file:

- **FILE ORDINARI: contengono indifferentemente testi, programmi simbolici, rilocabili assoluti, dati e qualsiasi altro tipo di informazione. Il loro contenuto non ha alcun significato per il sistema operativo ma solo per i programmi applicativi che li usano.**
- **DIRECTORY: è una collezione di nomi di file. Consente all'utente di organizzare i file in gruppi.**
- **FILE SPECIALI: tutte le periferiche sono trattate da UNIX come file speciali, il cui nome è proprio il nome della periferica. Associato a ciascuna periferica c'è un programma di sistema (driver) responsabile della comunicazione.**

FILE ORDINARI

OPERAZIONE	SIGNIFICATO
LETTURA (r)	Lettura del contenuto del file
SCRITTURA (w)	Modifica del contenuto del file
ESECUZIONE (x)	Esecuzione del file (il file deve contenere un progr. eseguibile)

DIRECTORY

OPERAZIONE	SIGNIFICATO
LETTURA (r)	Lista dei file nella directory
SCRITTURA (w)	Aggiunta o cancellazione di file nella directory
ESECUZIONE (x)	Spostamento nella directory

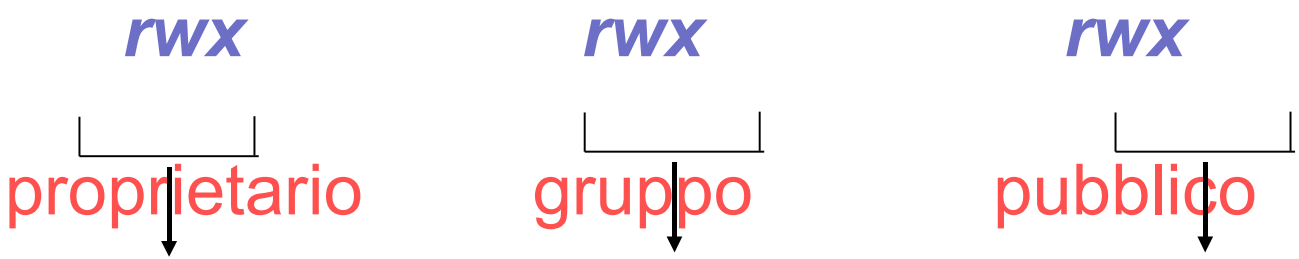
Ciascun file al momento della sua creazione possiede dei permessi di accesso per tre classi di utenti:

proprietario: colui che ha creato inizialmente il file

gruppo: eventuale gruppo a cui appartiene il proprietario

pubblico: tutti gli altri utenti

Su tutti i file è possibile permettere o negare le operazioni di lettura, scrittura ed esecuzione, mediante la stringa di 9 caratteri:



ESEMPI:

Se i permessi di un file ordinario eseguibile sono:

rwxr–xr – –

vuol dire che:

- il proprietario puo leggere, scrivere ed eseguire il file
- il gruppo può leggere ed eseguire il file ma non può modificarlo
- il pubblico può solo leggere il file

Se i permessi di una **directory** sono:

rwxr – xr – –

vuol dire che:

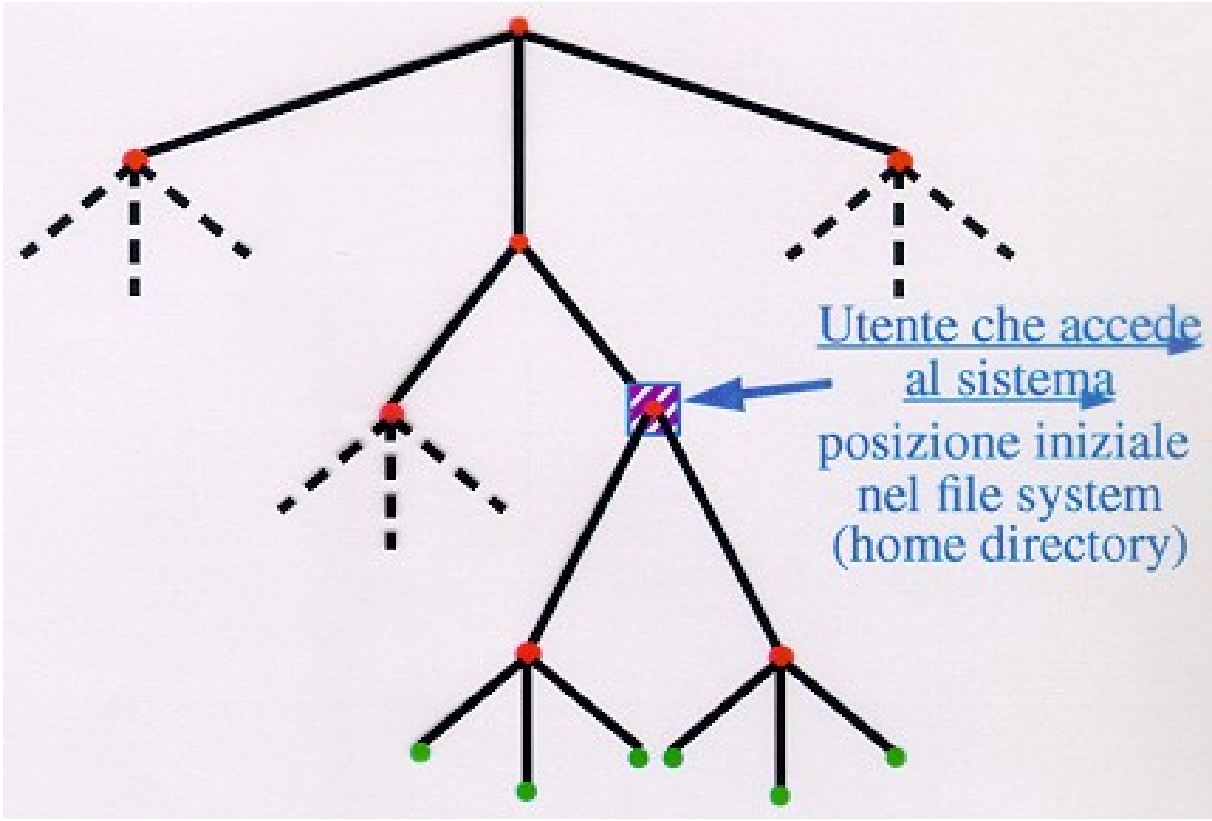
- il **proprietario** può vedere il contenuto della directory, può aggiungere o eliminare file, può spostarsi in quella directory

- il **gruppo** può vedere il contenuto della directory, non può aggiungere o eliminare file, può spostarsi in quella directory

- il **pubblico** può solo vedere il contenuto della directory.

Un **file** di sola lettura ha i seguenti permessi:

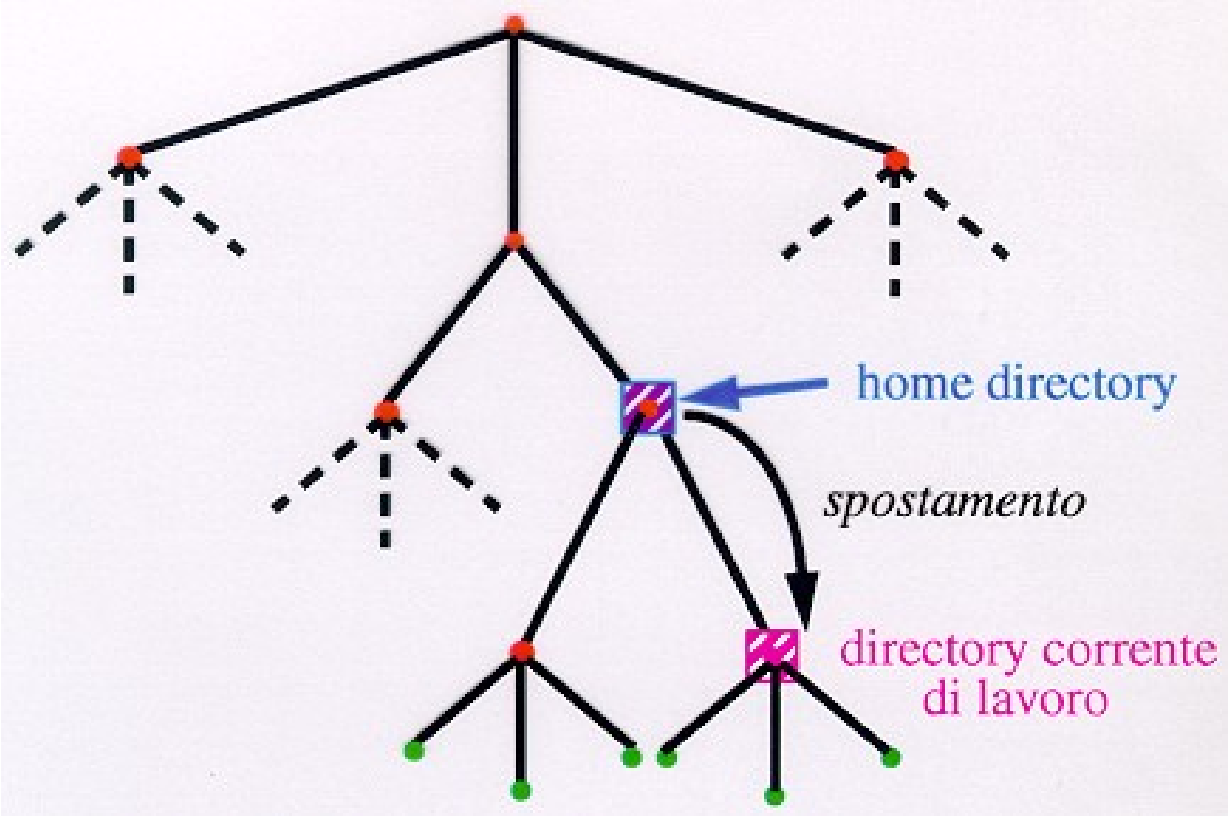
r – –r– –r – –



L'utente che accede al sistema è posizionato inizialmente in una directory prefissata detta

HOME DIRECTORY

Essa è sempre la stessa ogni volta che l'utente accede al sistema.

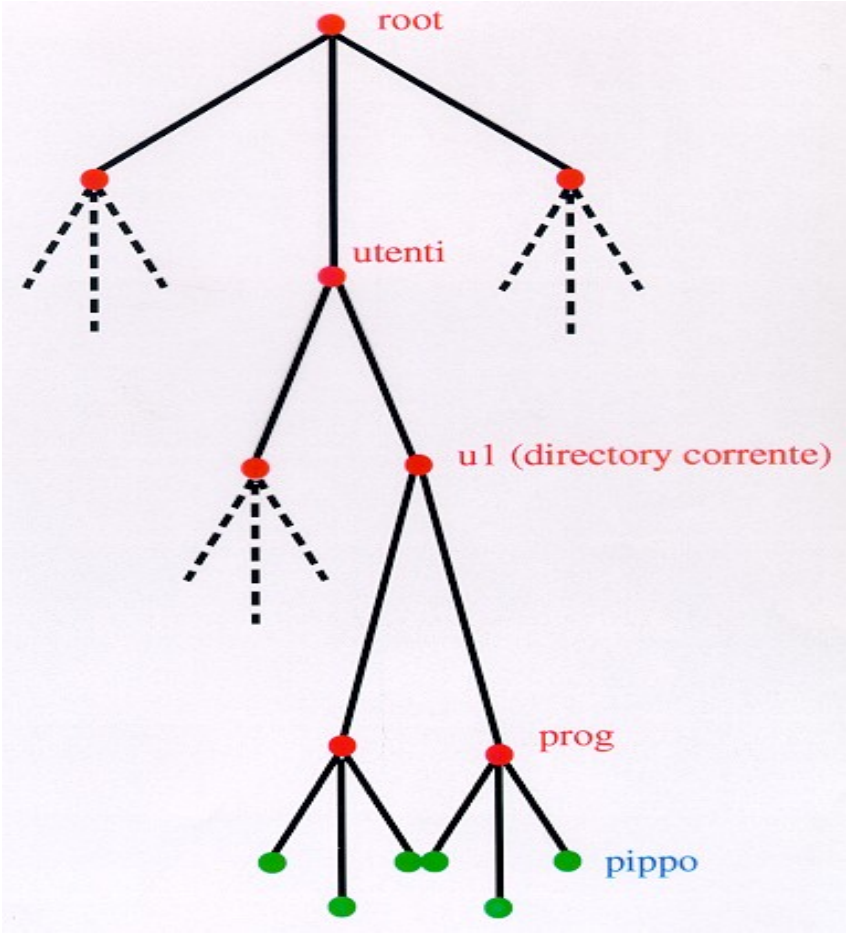


A partire dalla home directory l'utente può spostarsi in altre directory del file system. La

DIRECTORY CORRENTE DI LAVORO

è la directory dove un utente è posizionato ad un fissato istante.

IDENTIFICAZIONE DI FILE O DIRECTORY NEL FILE SYSTEM



COME FARE RIFERIMENTO AL FILE DI NOME pippo?

SOLUZIONE 1.

`/utenti/u1/prog/pippo`

Specificare il *cammino (path)* che occorre fare per andare *dalla radice del file sistem* (**root** indicata con /) fino al file pippo.

SOLUZIONE 2.

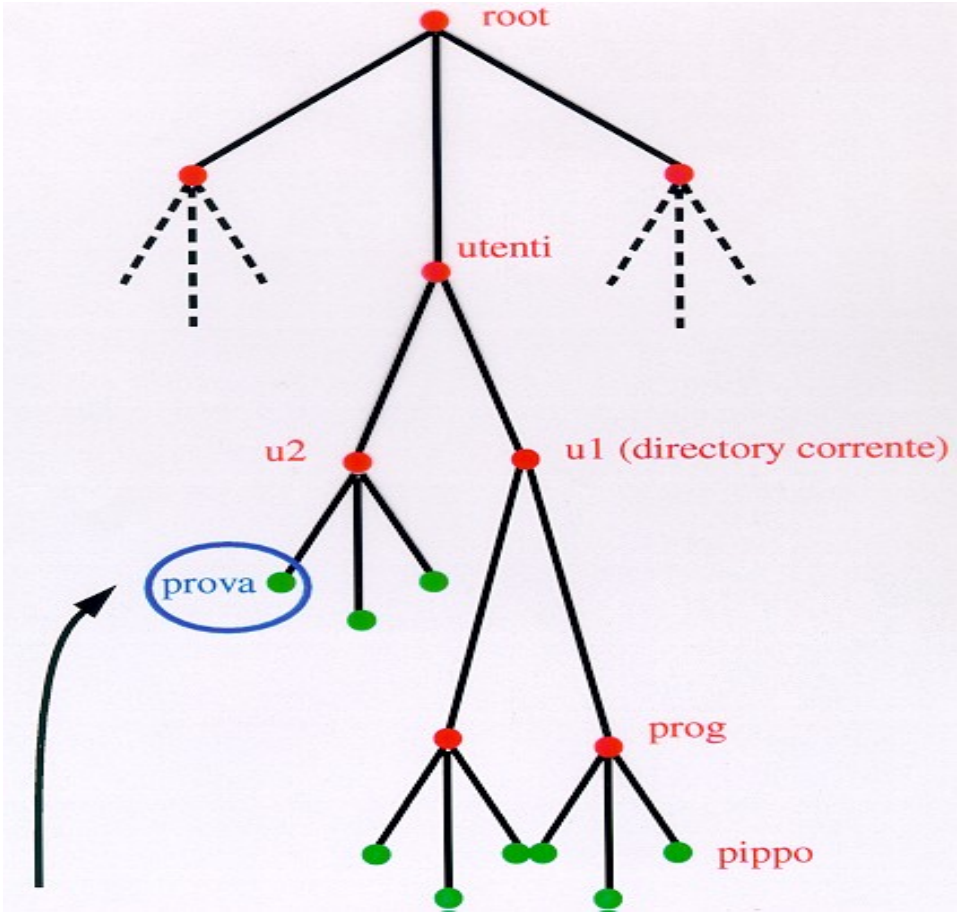
`prog/pippo`

Specificare il *cammino* che occorre fare per andare *dalla directory corrente* fino al file pippo *(path relativo)*

SOLUZIONE 3.

Spostarsi nella directory contenente il file pippo e fare riferimento ad esso *specificando semplicemente il suo nome*.

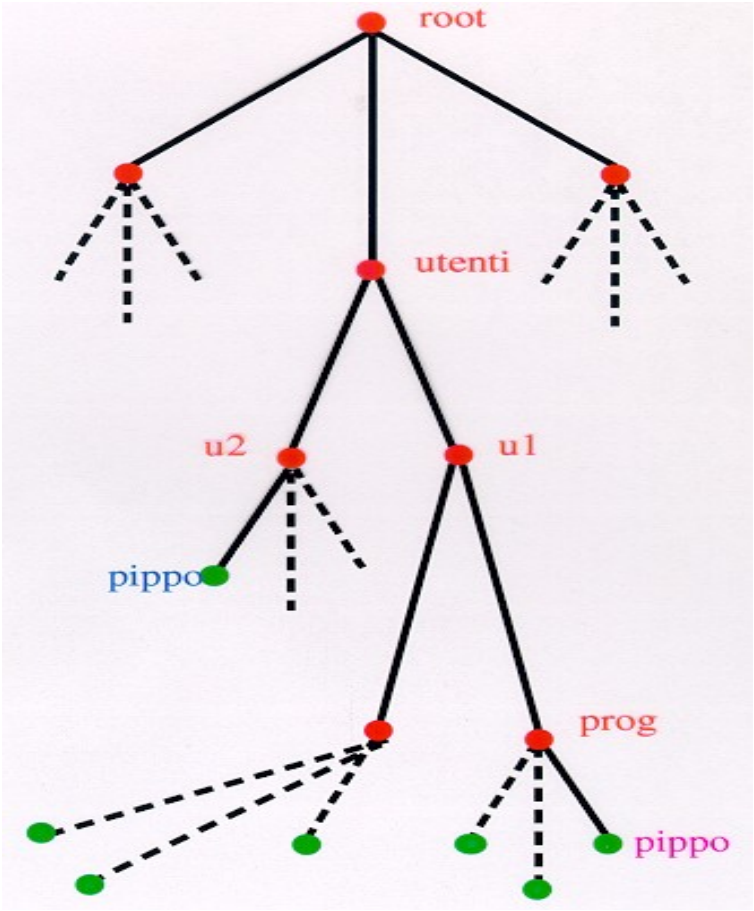
ESEMPIO



PATH ASSOLUTO: `/utenti/u2/prova`

PATH RELATIVO: `../u2/prova` (.. a directory padre)

ESEMPIO:



I due file **pippo** sono univocamente determinati dai path differenti:

`utenti/u2/pippo` `/utenti/u1/prog/pippo`

PRINCIPALI COMANDI

DI UNIX

(...e UNIX-like...)

SINTASSI DEI COMANDI

comando= [-opzioni] [argomento] [argomento] **..**

comando = nome di un file eseguibile nel file system. Il file può contenere sia un programma di sistema che un programma utente

opzioni = particolari parametri che specificano capacità opzionali del comando

argomento = operandi (in genere nomi di file)

Alcuni COMANDI

:: man *comando*
manual – *Visualizza alcune pagine di documentazione del comando specificato.*

:: passwd
password – *Comando interattivo di definizione o modifica della password (è richiesto l'inserimento della nuova password per due volte affinchè la modifica sia accettata)*

::pwd
print work directory – *Visualizza il path assoluto della directory corrente di lavoro*

::who
elenca le user id degli utenti collegati in quel momento al sistema

::mail *utente*
Spedisce posta all'utente specificato

ESEMPIO:

L'utente caio si collega al sistema. Vuole cambiare la sua password:

```
matna3.dma.unina.it$passwd
Changing password for caio
Old password: (digitare la vecchia password)
Enter new password: (digitare la nuova password)
Verify: (digitare di nuovo la nuova password)
matna3.dma.unina.it$
```

ESEMPIO:

L'utente ciao vuole sapere la sua directory corrente di lavoro:

```
matna3.dma.unina.it$pwd
/usr/users/caio
matna3.dma.unina.it$
```

L'utente ciao vuole r sapere chi è collegato al sistema:

```
matna3.dma.unina.it$ who
tizio ttypf Mar 6 15:24 (193.204.16.30)
caio ttyq0 Mar 6 15:53 (193.167.16.17)
sem ttyq1 Mar 6 16:03 (193.204.16.1)
```

COMANDI DI GESTIONE FILE

- **ls** path

list – richiesta di listare il contenuto di path (se path non è specificato , viene listato il contenuto della directory corrente di lavoro)

- **mkdir** path

make directory – crea una directory come specificato dal path.

- **cd** path

change work directory – cambia la directory corrente di lavoro in quella specificata da path

- **cat** path

concatenate – lista il contenuto del file specificato da path sul supporto standard di output (schermo)

- **pg** path

page – ha lo stesso significato di cat ma l'output viene fornito su più pagine, dove una pagina è lo schermo del terminale.

- **cp** path1 path2

copy – copia il file specificato da path1 nel file specificato da path2. Se il secondo argomento è una directory, il file è copiato in questa directory con lo stesso nome

- **lp -d**dest file

line printer spooler – invia il file alla stampante specificata in dest.

- **rm** path

remove – cancella il file specificato da path

- **rmdir** path

remove directory – cancella la directory specificata da path. Una directory può essere cancellata solo se è vuota

- **mv** path1 path2

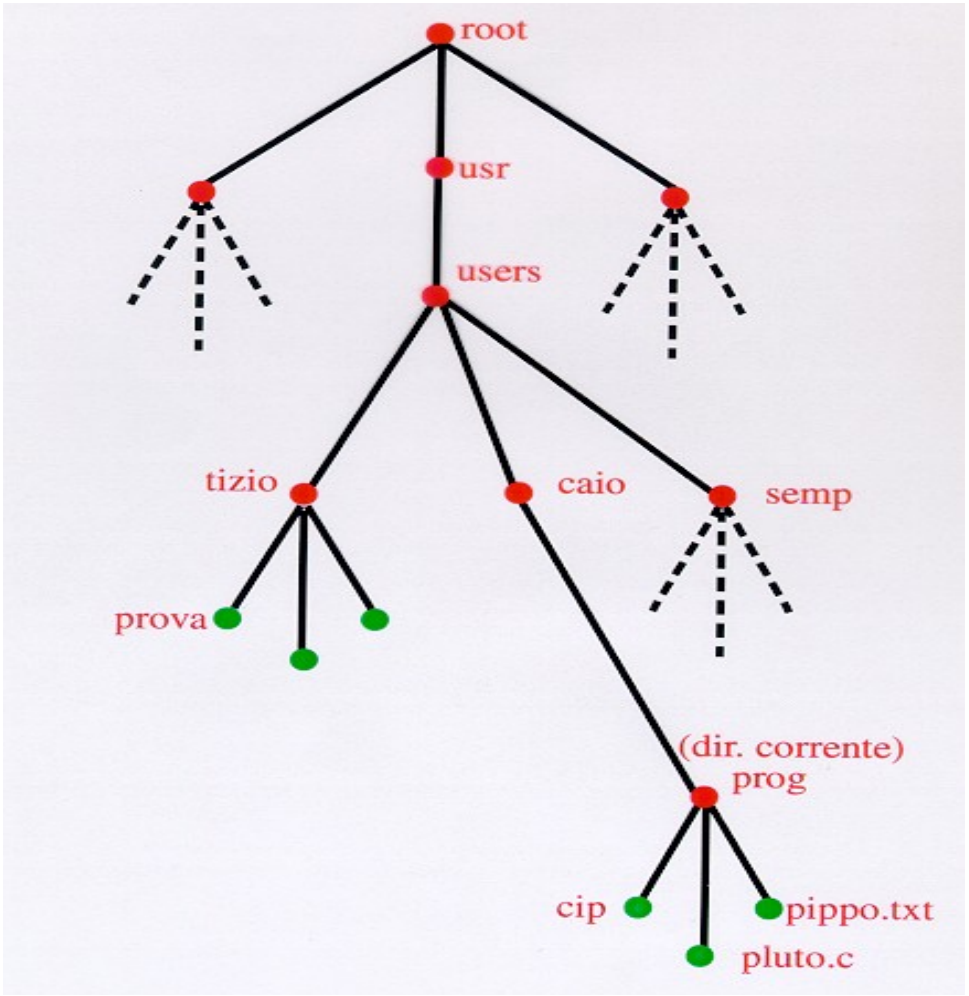
move – muove il contenuto del file specificato da path1, nel file specificato da path2.

- **cc** file1 file2…filen

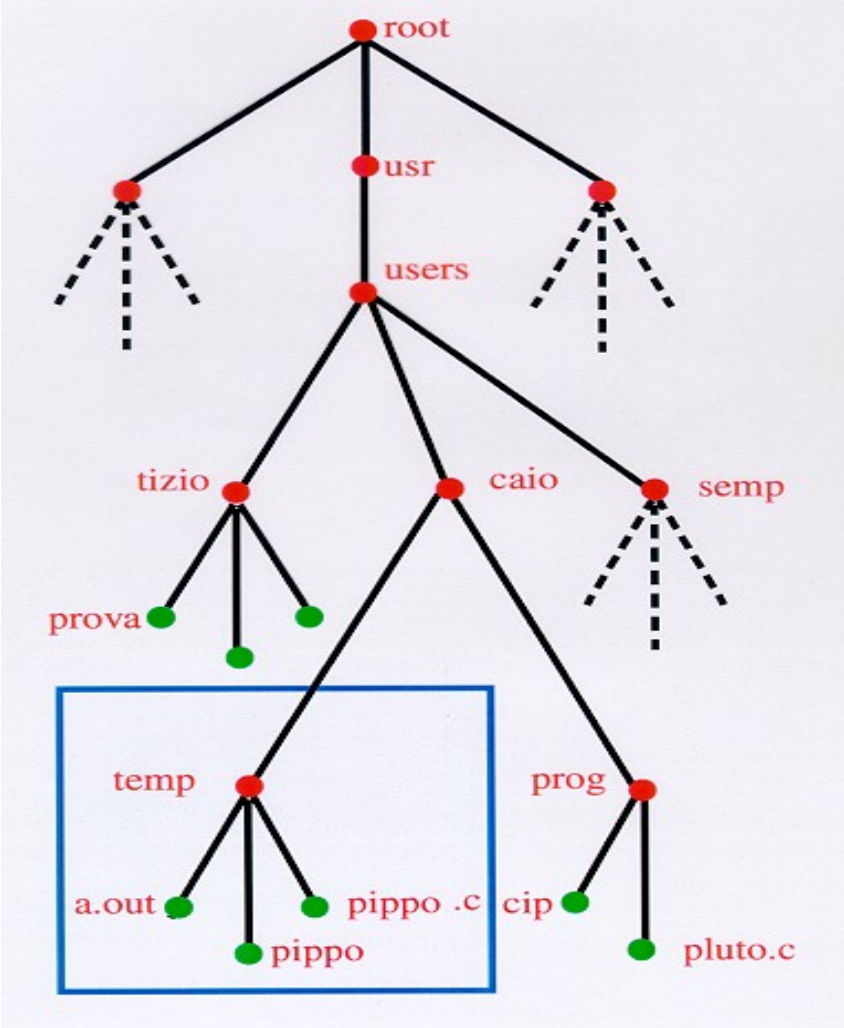
compilatore C – viene richiamato il compilatore C, che a partire da file1 file2 … filen produce il programma eseguibile (assoluto) di nome standard a.out.

¡

ESEMPIO 1:
Situazione attuale



Cosa si vorrebbe ottenere



Realizzazione

```
matna3.dma.unina.it$ pwd
/usr/users/caio/prog
matna3.dma.unina.it$
```

1. creare la directory temp

```
matna3.dma.unina.it$ mkdir ../temp
```

2. spostare il file pippo .txt, chiamandolo pippo.c, nella directory temp

```
matna3.dma.unina.it$
                    mv pippo.txt ../temp/pippo.c
```

3. spostarsi nella directory temp e chiamare il compilatore C per creare a.out (eseguibile di pippo.c)

```
matna3.dma.unina.it$ cd ../temp
matna3.dma.unina.it$ pwd
/usr/users/caio/temp
matna3.dma.unina.it$ cc pippo.c
matna3.dma.unina.it4 ls
a.out
pippo.c
```

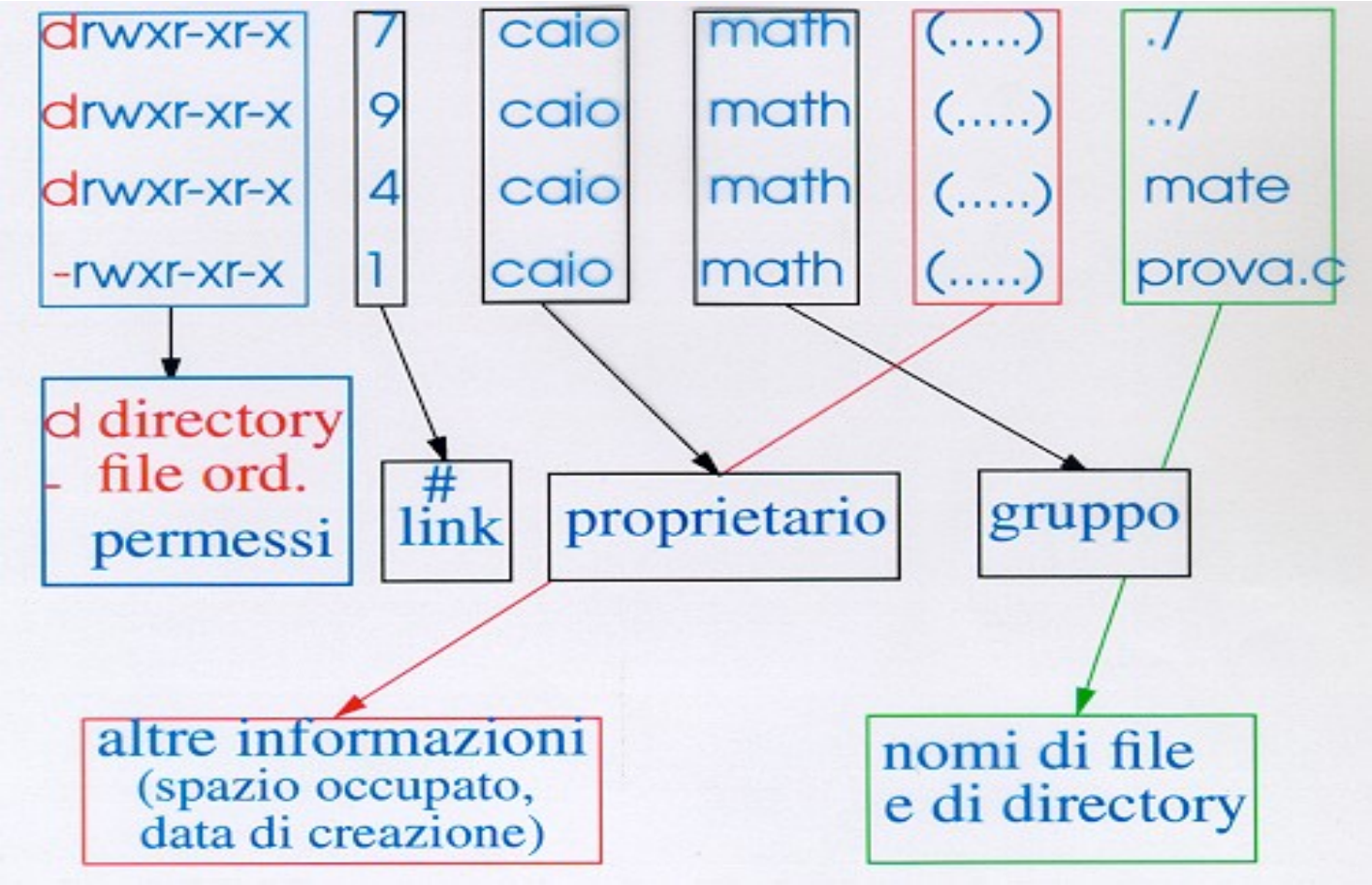
4. creare l'eseguibile di pippo.c di nome pippo

```
matna3.dma.unina.it$ cc -o pippo pippo.c
matna3.dma.unina.it$ ls
a.out
pippo.c
pippo
matna.dma.unina.it$
```

ESEMPIO 2:

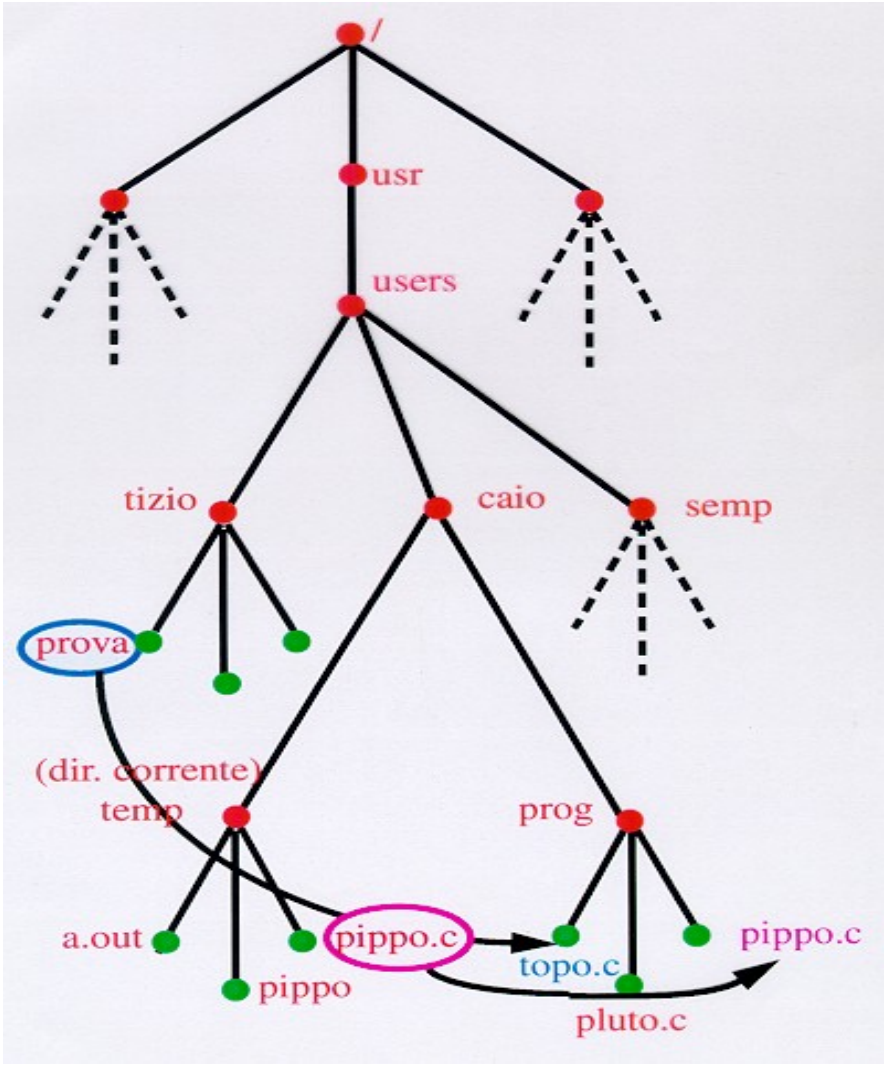
Listare il contenuto di una directory in modo da ottenere il maggior numero di informazioni.

```
matna3.dma.unina.it$ ls -la
```



ESEMPIO 2:

Copiare i file cerchiati come indicato dalle frecce



Come Fare

matna3.dma.unina.it$ **pwd**
/usr/users/caio/prog
matna3.dma.unina.it$

1. spostarsi nella home directory

matna.dma.unina.it$ cd
matna3.dma.unina.it$

2. copiare il file prova, chiamandolo topo.c, nella directory temp

matna3.dma.unina.it$ **cp ../tizio/prova prog/topo.c**

COME FARE (CONT.)

3. copiare il file pippo.c nella directory prog

```
matna3.dma.unina.it$ cp temp/pippo.cprog
matna3.dma.unina.it$ ls prog
pippo.c
pluto.c
topo.c
matna.dma.unina.it$
```

STANDARD I/O

Nell'esecuzione di un comando entrano in gioco tre file standard:

Standard input file dal quale il comando legge il suo input (per default la tastiera).

Standard output file sul quale il comando scrive il suo output (per default il video)

Standard error file sul quale il comando scrive eventuali messaggi di errore (per default il video)

È possibile dirigere temporaneamente l'input e l'output verso altri file con l'uso dei caratteri:

< input > **o** >> output

scrive nel file **appende al file**

ESEMPIO 4:

Problema:

Eseguire il programma assoluto pippo, stampare il programma sorgente e i risultati sulla stampante lpribm.

Soluzione:

matna3.dma.unina.it$ **cd temp**

1. Esecuzione del programma pippo e redirezione dell'output al file results

matna3.dma.unina.it$ **pippo > results**

2. Il file results è appeso al file pippo.c

matna3.dma.unina.it$ **cat results >> pippo.c**

3. Il file pippo.c viene stampato

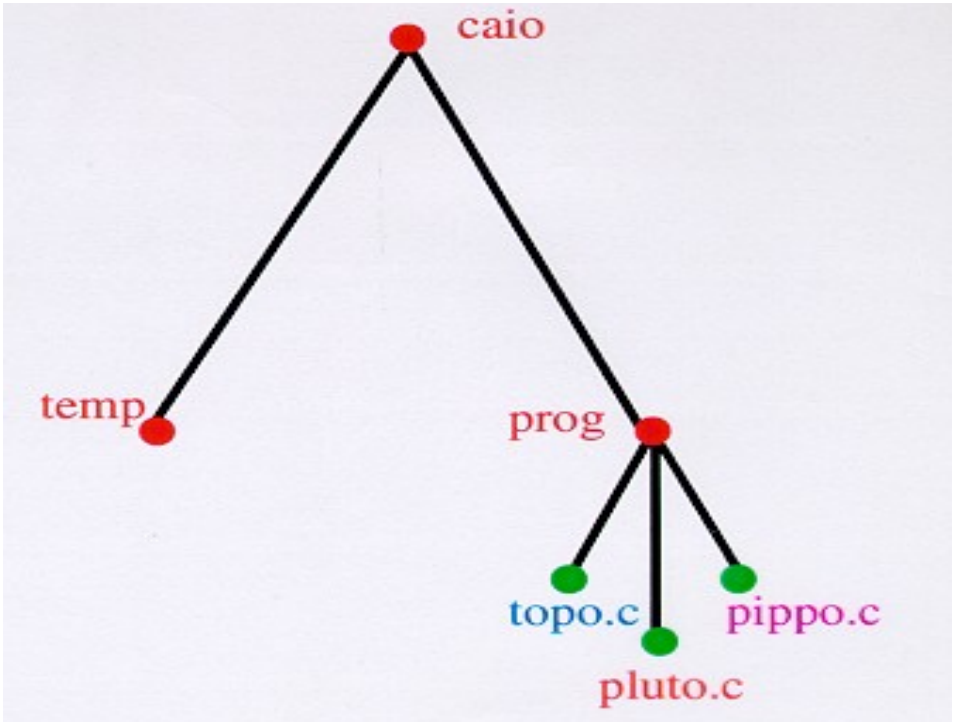
matna 3.dma.unina.it$ **lp -dlpribm pippo.c**

ESEMPIO 5:

Svuotare la directory temp senza eliminarla.

```
matna3.dma.unina.it$ pwd
/usr/users/caio
matna3.dma.unina.it$ rm temp/*.*
matna3.dma.unina.it$ ls temp
matna3.dma.unina.it$
```

Situazione dopo l'esecuzione dei comandi:



Comandi base (Riepilogo)

- ***Operazioni comuni***
  - **Lista dei files nella directory corrente**
    > **`ls`** |RETURN|
  - **Lista dei files il cui nome termina con .txt nella directory corrente**
    > **`ls *.txt`** |RETURN|
  - **Stampa della dimensione e delle proprietà dei files**
    > **`ls -l`** |RETURN|

```
> -rwxr-xr-x    1 desalvo  xv          11890 Jun  4  2001 test
> -rw-r--r--    1 desalvo  xv            148 Jun  4  2001 test.c
> drwxr-xr-x    2 desalvo  xv           4096 Jan 15 22:09 test-directory
```

  - **Stampa del nome della directory di lavoro corrente (full pathname)**
    > **`pwd`** |RETURN|
  - **Cambio di directory**
    > **`cd <nome directory>`** |RETURN|
  - **Ritorno alla directory al livello precedente**
    > **`cd ..`** |RETURN|
  - **Creazione di una nuova directory**
    > **`mkdir <nome directory>`** |RETURN|
  - **Rimozione di una directory vuota**
    > **`rmdir <nome directory>`** |RETURN|

Comandi base (Riepilogo) (cont)

- ***Operazioni comuni (2)***
  - **Spostamento o rinominazione di files**
    - > **`mv <file originale> <nuovo file>`** |RETURN|
    - > Esempio di spostamento: **`mv pippo.txt txt_files/`**
    - > Esempio di rinominazione: **`mv pippo.txt paperino.txt`**
  - **Cancellazione di uno o più files**
    - > **`rm <nome file>`** |RETURN|
    - > **`rm pippo*`** |RETURN|
  - **Copia di un file**
    - > **`cp pippo.txt paperino.txt`** |RETURN|
  - **Visualizzazione del contenuto di un file (browsing)**
    - > **`more pippo.txt`** |RETURN|
  - **Creazione o modifica di un file di testo (editing)**
    - > **`emacs pippo.txt`** |RETURN|
  - **Stampa di un file**
    - > **`lpr -P<nome stampante> <nome file>`** |RETURN|
  - **Informazioni sull'uso di un comando**
    - > **`man <comando>`** |RETURN|

Accesso al sistema

- ***Ogni utente è identificato da una coppia di informazioni***
  - **Nome dell'utenza (*username*), pubblica**
  - **Parola-chiave (*password*), segreta**

- ***L'utente accede al sistema fornendo username e password***
  - **Login: accesso al sistema (da remoto)**
    - `ssh <nome del computer>`
  - **Logout: uscita dal sistema**
    - `exit`

- ***Più utenti possono lavorare contemporaneamente sulla stessa macchina***

Una nota sui

Sistemi Opertivi

UNIX-like

(...con sorgenti aperti...)

Linux

- ***Linus Torvalds, studente dell'università di Helsinki, inizia per hobby nel 1991 la scrittura di un'estensione del mini S.O. UNIX di nome Minix***
- ***Nel 1994 la prima versione del kernel di Linux vede la luce***
- ***Linux è sviluppato secondo la licenza GNU General Public License (GPL) e tutto il codice sorgente è distribuito gratuitamente al pubblico***
- ***Le performances e la versatilità di Linux lo hanno fatto diventare ben presto un sistema operativo di grande diffusione***
- ***Linux ha introdotto l'idea di OpenSource***
  - **Accesso al codice sorgente**
  - **Redistribuzione gratuita del codice e dei programmi compilati**
  - **Permesso di effettuare modifiche**
  - **Preservazione dell'integrità del codice sorgente dell'autore**

OpenSource

- ***Apertura del codice***
  - ***Avere a disposizione il codice sorgente agevola la risoluzione di problemi***
- ***Stabilità***
  - ***Dal momento che l'OpenSource può contare su una grande comunità di sviluppatori e che tutto il codice sorgente sviluppato deve rimanere pubblico, il software Open Source è generalmente più stabile del software commerciale***
- ***Adattabilità***
  - ***Il codice OpenSource è strettamente collegato con delle regole standard (Open Standards) e quindi è facilmente adattabile ad ogni altra applicazione OpenSource***
- ***Qualità***
  - ***I prodotti OpenSource possono essere testati da una grande comunità, che è anche potenzialmente una comunità di sviluppatori***
- ***Innovazione***
  - ***Ciò che spinge l'OpenSource è la competizione e la possibilità di contribuire comunemente allo sviluppo di un prodotto***
- ***Costo***
  - ***La maggior parte del software OpenSource è gratis (ma non lo è necessariamente)***