Corso Linux Base

2. Uso del Sistema

Logging In (1)

Una volta partito il sistema l'utente esegue la procedura di Login. Se si è installato un ambiente grafico di desktop, la procedura è gestita dall'ambiente stesso
Il *Login* richiede uno *username* e una *password*

Logging In (2)

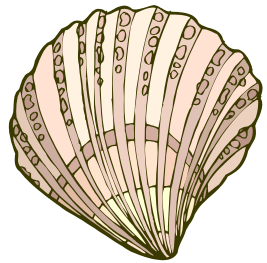
Al momento della registrazione, all'utente viene assegnato anche una zona di disco identificata come home directory
Alla fine della procedura di *Login* l'utente "si trova" nella sua *home directory* cioè in una zona di disco in cui ha tutti i privilegi per creare nuovi *files* o *directories*
(v. in seguito per esempi)

Logging In (3)

- Sono stati preparati degli account :
  lclbxx xx=01...50
  password lclbxx
- Es: Username lclb07
  Password lclb07

Uso della Shell (1)

- In Linux (e in UNIX) si interagisce con il sistema attraverso un programma chiamato shell che viene invocato automaticamente al *Login*



- Esistono diverse *shell* (**`sh`**, **`bash`**, **`tcsh`**, …)
  - la maggior parte dei comandi ha la stessa sintassi
  - la scelta di una *shell* è essenzialmente questione di gusti

Uso della Shell (2)

In Linux i comandi sono dati sotto la *shell* che li interpreta, esegue e scrive (se del caso) il risultato su terminale

L'attenzione della shell è espressa dal *prompt* , carattere che può essere deciso dall'utente

Esempi di *prompt* :

```
>
%
[pccmslb]:~>
```

Uso della Shell (3)

I comandi Linux (e UNIX) sono stringhe di caratteri molto sintetiche. In tempi recenti sono state aggiunte interfacce grafiche per facilitare alcune operazioni. Qui trattiamo principalmente la versione testuale.
Ogni comando ha un insieme di opzioni che possono essere aggiunte in linea come singole lettere minuscole e/o maiuscole precedute dal segno “-” (vedi in seguito)

Comandi di uso comune

Il comando di *help* non si chiama **`help`** !

Per avere informazioni su un qualsiasi comando si usa **`man`** che mostra l'uso del comando e delle sue opzioni in modo formattato

Ovviamente si può avere anche

```
> man man
```

che illustra il funzionamento del comando **`man`**

Comandi di uso comune: `ls`

- Il comando **`ls`** mostra la lista di tutti i file contenuti in una directory

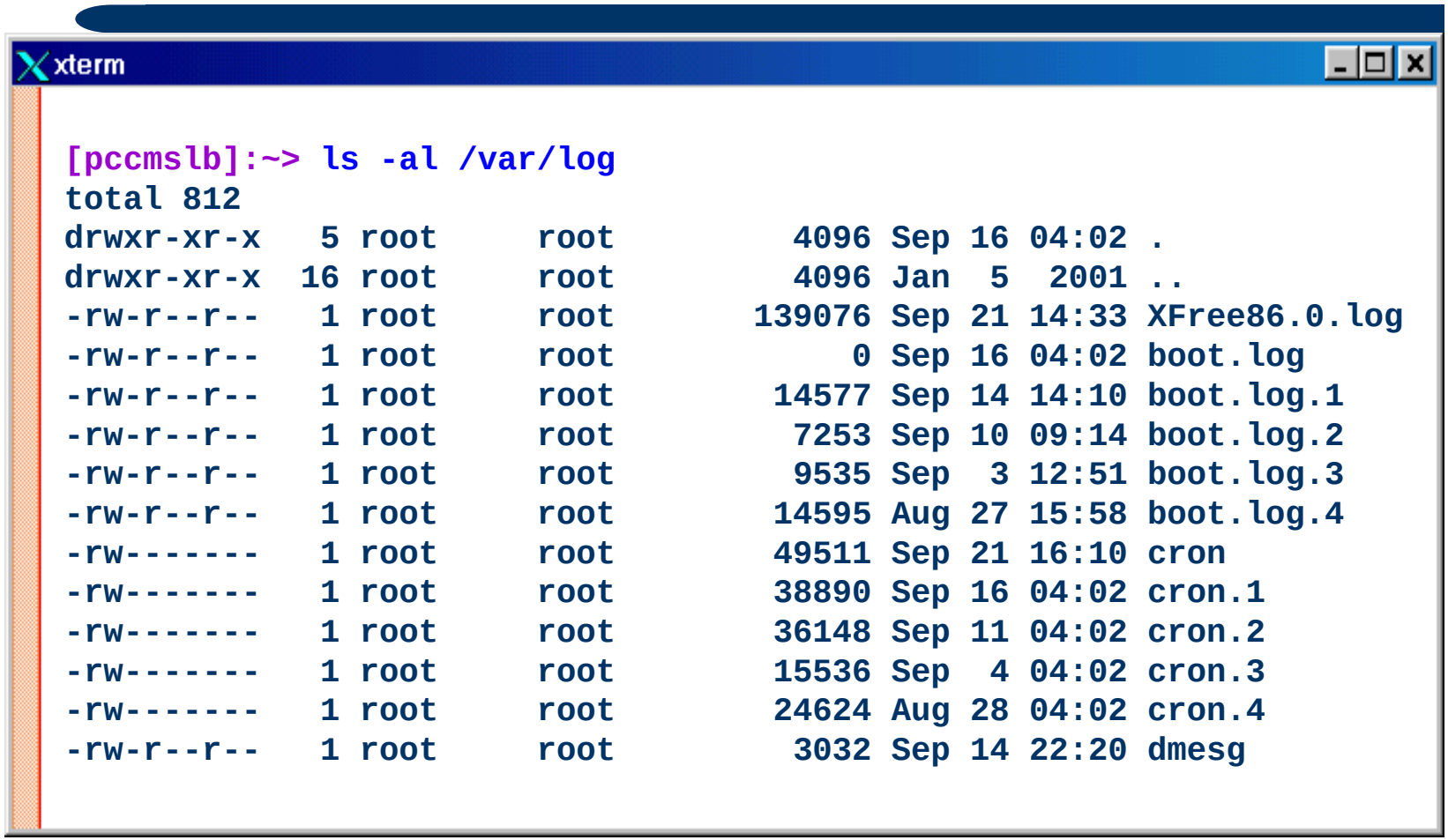
  `> ls`

  mostra i file nella directory corrente

  `> ls /dir`

  mostra i file nella directory **`/dir`**

Comandi base: ls



Comandi base: `cp`

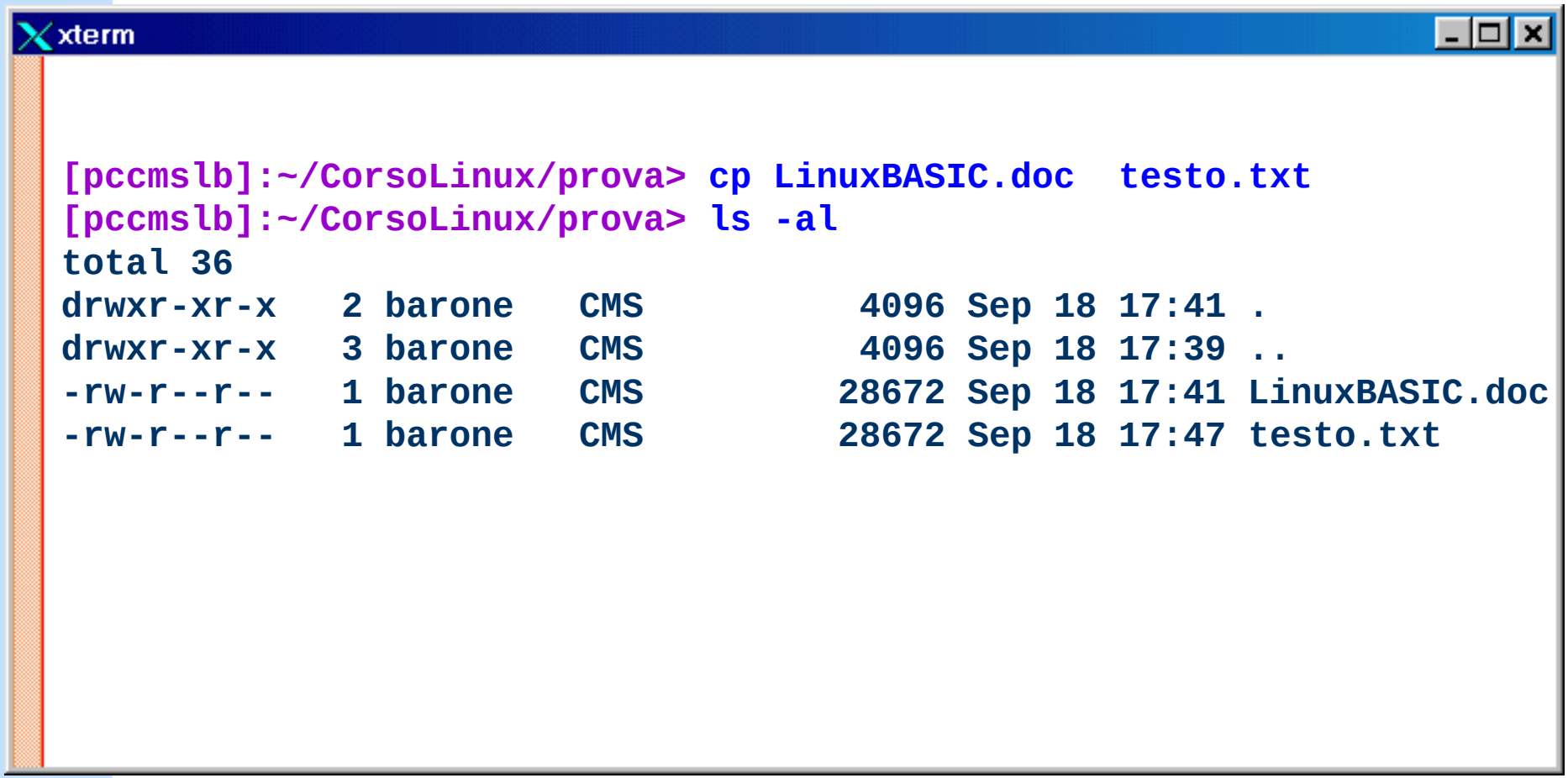
Il comando **`cp`** serve a copiare uno o più *file*s da una locazione a un'altra

La sintassi tipica è

```
> cp <inputFile> <outputPath>
```

Nell'esempio copio il file **`LinuxBASIC.doc`** dalla directory corrente cambiando il nome del file

Esempio


xterm
[pccmslb]:~/CorsoLinux/prova> cp LinuxBASIC.doc testo.txt
[pccmslb]:~/CorsoLinux/prova> ls -al
total 36
drwxr-xr-x   2 barone   CMS   4096 Sep 18 17:41 .
drwxr-xr-x   3 barone   CMS   4096 Sep 18 17:39 ..
-rw-r--r--   1 barone   CMS   28672 Sep 18 17:41 LinuxBASIC.doc
-rw-r--r--   1 barone   CMS   28672 Sep 18 17:47 testo.txt

Comandi base: `mv`, `rm`

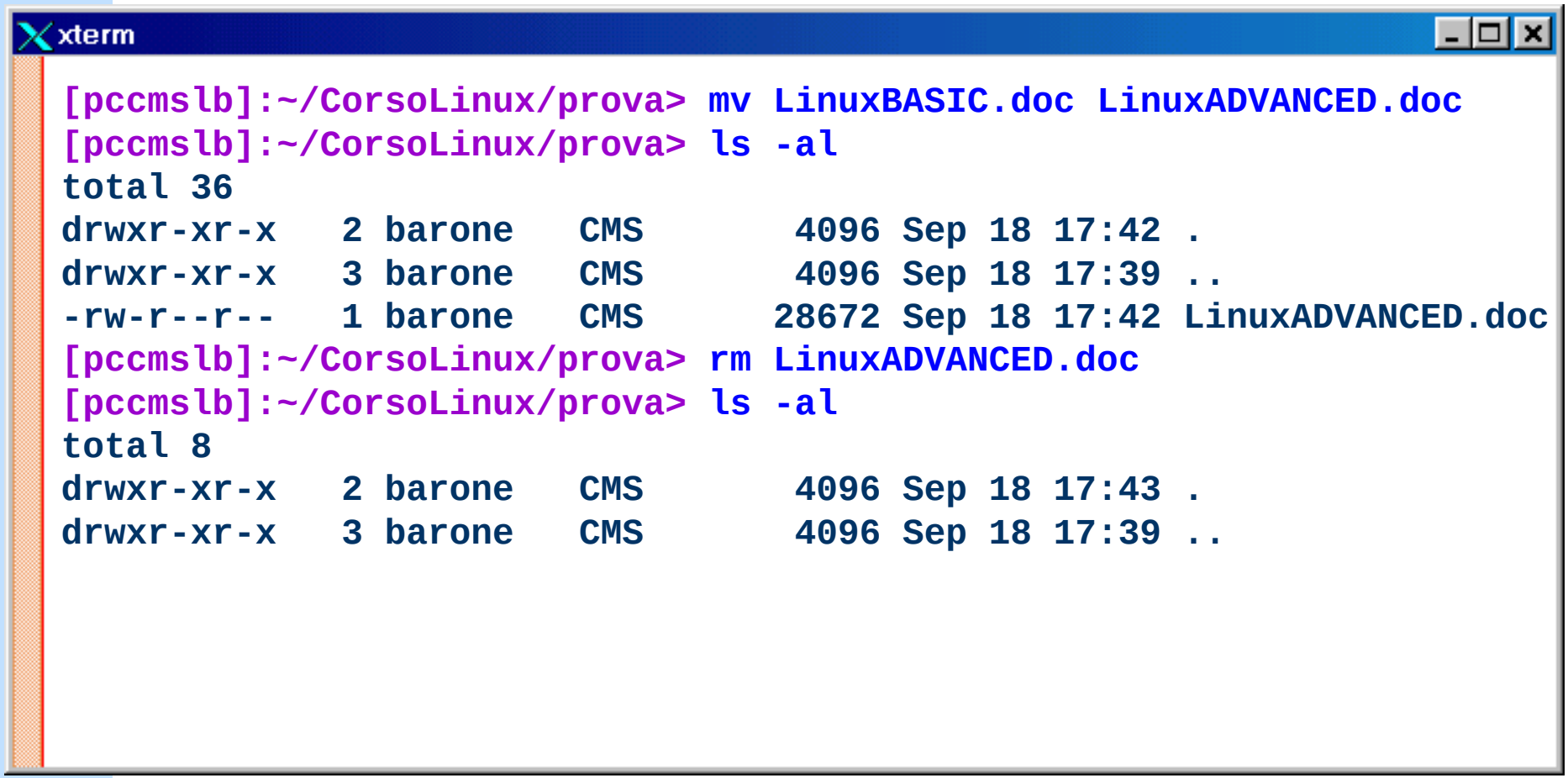
Il comando **`mv`** (da *move*) serve a cambiare nome a un file o a una directory

```
> mv <vecchioNome> <nuovoNome>
```

Il comando **`rm`** (da *remove*) serve a cancellare un file o una directory

```
> rm <file>
```

Esempio


xterm
[pccmslb]:~/CorsoLinux/prova> mv LinuxBASIC.doc LinuxADVANCED.doc
[pccmslb]:~/CorsoLinux/prova> ls -al
total 36
drwxr-xr-x   2 barone   CMS   4096 Sep 18 17:42 .
drwxr-xr-x   3 barone   CMS   4096 Sep 18 17:39 ..
-rw-r--r--   1 barone   CMS   28672 Sep 18 17:42 LinuxADVANCED.doc
[pccmslb]:~/CorsoLinux/prova> rm LinuxADVANCED.doc
[pccmslb]:~/CorsoLinux/prova> ls -al
total 8
drwxr-xr-x   2 barone   CMS   4096 Sep 18 17:43 .
drwxr-xr-x   3 barone   CMS   4096 Sep 18 17:39 ..

Dov’è lo START Button?

- Gli utenti Windows sono spesso turbati dall’uso della shell, perché abituati ad utilizzare dei menu
- La shell è un mezzo molto potente, che consente grande efficienza e libertà, ma se si desidera utilizzare il PC in stile Windows, si possono utilizzare i tools grafici dei Window Managers come GNOME e KDE che sono analoghi a quelli di Windows

Come creo il mio programma? (1)

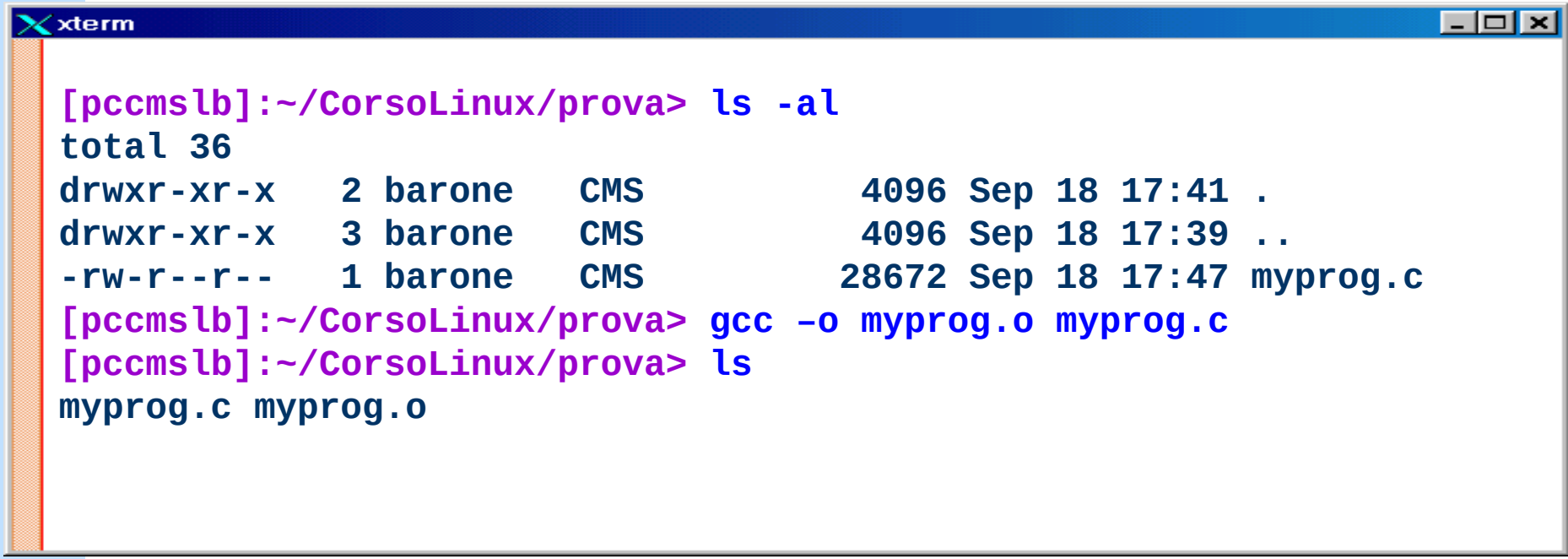
- Apro un *editor* per scrivere le istruzioni e alla fine salvo il file con un nome (v.dopo)
- Devo compilare il programma: il compilatore
  - legge le istruzioni da me scritte
  - controlla la sintassi
  - se la sintassi è corretta traduce le istruzioni di alto livello in basso livello (codice oggetto) producendo un nuovo file

Come creo il mio programma? (2)

- Ma compilare non basta...
  Bisogna “agganciare” (link) le librerie di sistema
- ... e se necessario (ad es. uso funzioni matematiche) “agganciare” altre librerie
- Ora il programma è pronto per l’esecuzione

Come si compila ?

- Avendo editato e salvato un file come **myprogram.c** uso il comando **gcc (o cc)**

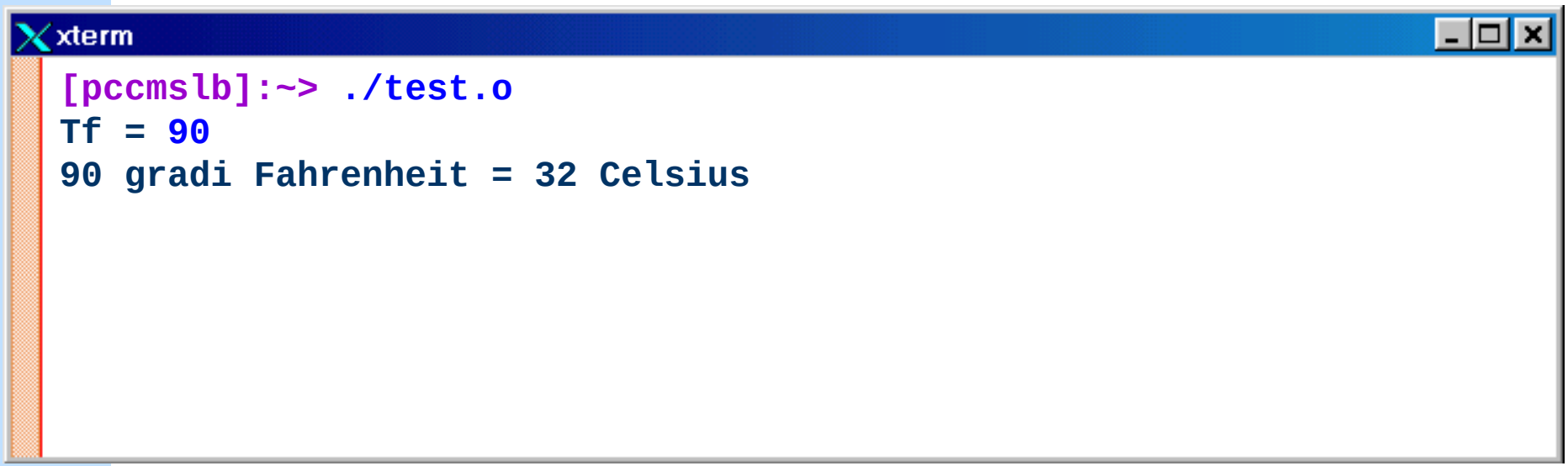


ATTENZIONE

- Se non usate l'opzione **`-o`** il codice oggetto verrà scritto SEMPRE in **`a.out`**
- L'operazione di *link* base è svolta automaticamente da **`gcc`**
- Se occorre la libreria matematica si aggiunge l'opzione **`-lm`** .
  Es: **`gcc -lm -o test.o myprog.c`**

Esecuzione

Per eseguire un programma è sufficiente darne il nome, prefissato da ./



Editor Principali

- I principali programmi di *editor* sono:
  - vi (sempre presente nella installazione base)
  - emacs
  - nedit (non sempre disponibile nella installazione base, si può scaricare da **`http://www.nedit.org`**)
- Usiamo **`emacs`**

Uso di **emacs**

- Per aprire l'editor su un file nuovo a cui diamo il nome **program.c**

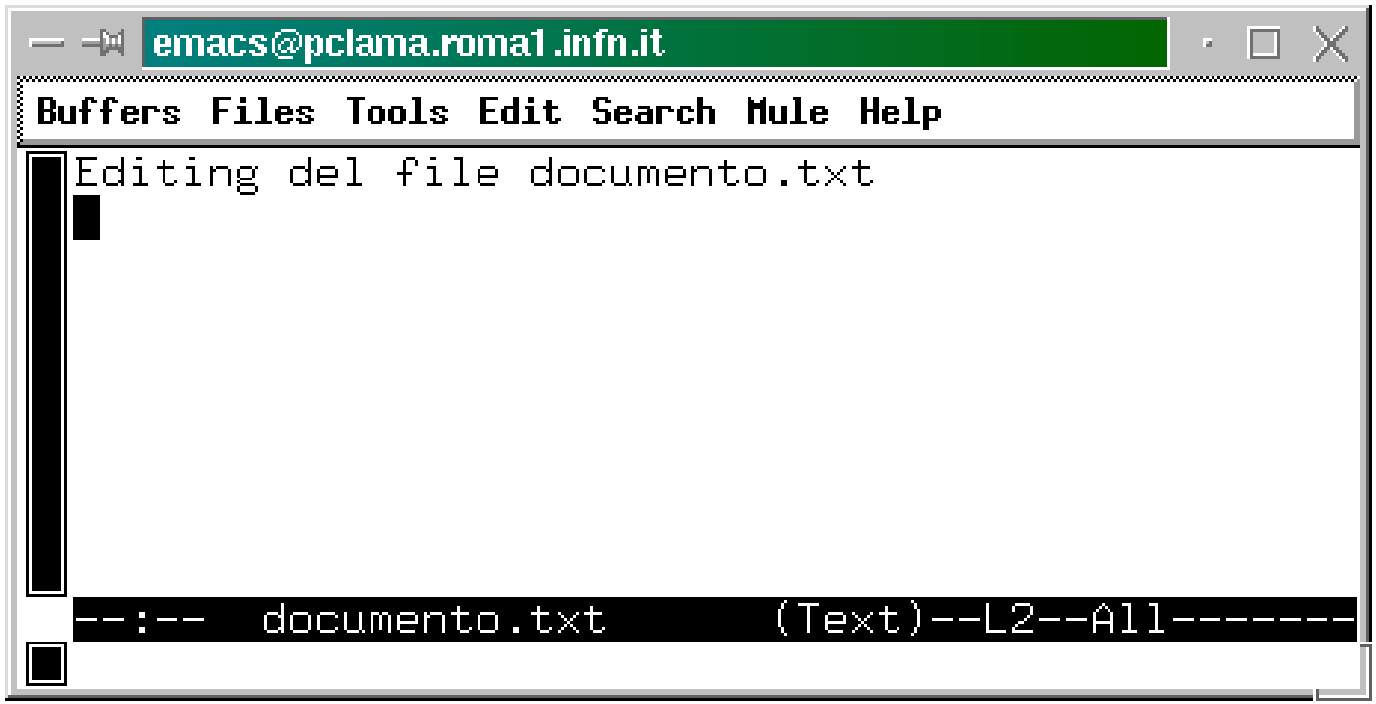
```
> emacs program.c&
```

notate il simbolo **&** : questo permette di aprire l'editor e poi mantenere il controllo del prompt

```
>
```

Uso di emacs (2)

- Appare una finestra di questo tipo



Uso di **emacs** (3)

- Scrivere il programma e salvare usando il menu a tendina o il comando ctrl-x ctrl-s (premere contemporaneamente i tasti ctrl e x e poi i tasti ctrl e s)
- Se si è usato il carattere **&** si può compilare senza uscire dall'editor, spostandosi nella finestra della shell