Laboratorio di Informatica di Base

Progetto Tandem 2007/2008

Docente: *Carlo Drioli*

Lucidi a cura di

Andrea Colombari, (colombari@sci.univr.it)

Carlo Drioli (drioli@sci.univr.it) e

Barbara Oliboni (oliboni@sci.univr.it)

Lezione 1

Elaborazione di testi

Tool di elaborazione testi

- In Linux è particolarmente importante disporre di strumenti efficaci per poter leggere, modificare e scrivere file di testo.
- Molte operazioni di **configurazione e manutenzione** del sistema richiedono la modifica di file testuali.
- I programmi di elaborazione di file di testo storici in Linux sono ***vi*** ed ***emacs***. Ne esistono molti altri, fra cui si ricordano ***joe, kate, nano***.

vi: caratteristiche

- *vi* è un editor di tipo “screen editor”, il terminale viene cioè usato per visualizzare una pagina di testo.
- E’ possibile spostare il cursore nel file e fornire comandi tramite combinazioni di tasti.
- Può operare in una delle seguenti modalità per volta: **comando**, **testo**, o **editor di linea**.

joe: caratteristiche

- ***joe*** è, come *vi,* uno “screen editor”.
- E’ possibile spostare il cursore nel file e fornire comandi tendendo premuto il tasto **CTRL** e combinandolo con altri tasti.

emacs: caratteristiche

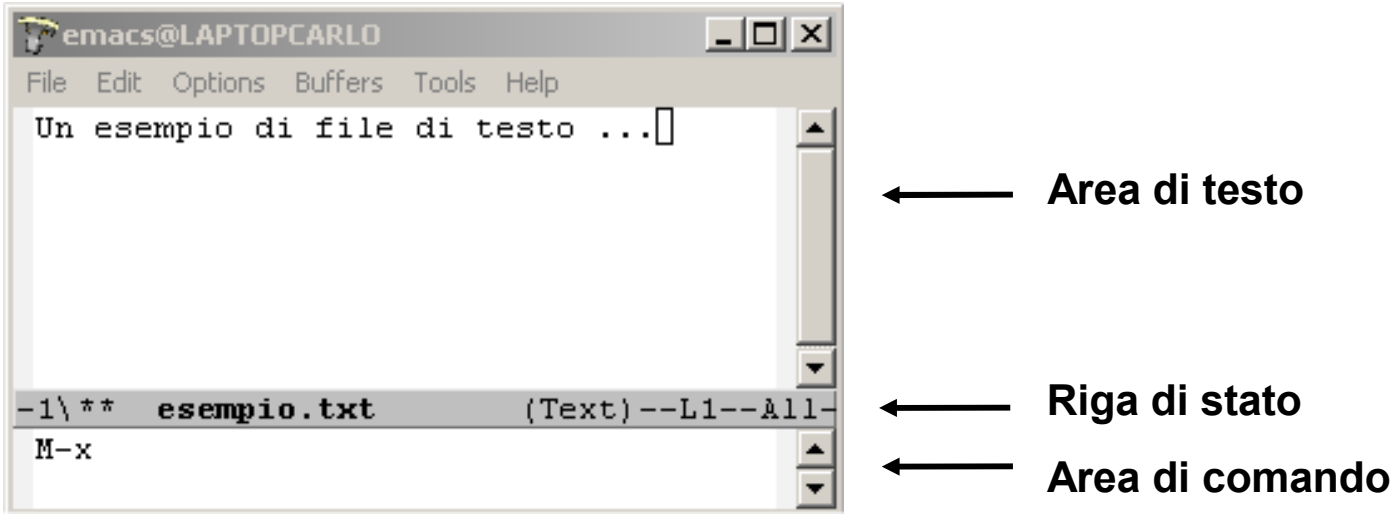
- In ***emacs*** non esistono modalità distinte di funzionamento come in vi.
- I comandi sono invocati tramite combinazioni dei tasti **CTRL, ALT e ESC** con altri caratteri .

Elaborazione di testi con *emacs*

- Creare o modificare un documento

```
$ emacs nomefile
```

- Lo spazio dello screen editor è diviso in tre parti
  - Area di testo
  - Riga di stato
  - Area di comando

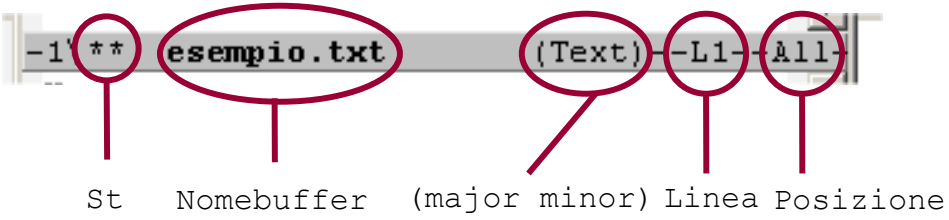


Elaborazione di testi con *emacs* (2)

- Emacs opera su tre componenti principali:
  - File: è un file memorizzato sul disco. Non viene mai manipolato direttamente, tutte le operazioni vengono eseguite copiando i file in dei buffer di memoria e salvando il risultato delle manipolazioni sui buffer in un file.
  - Buffer: è una struttura interna che contiene il testo da elaborare. Possono esserci più buffer attivi allo stesso tempo.
  - Finestre: una finestra corrisponde alla visualizzazione di un buffer. E' possibile visualizzare uno o più buffer per volta aprendo e chiudendo finestre durante una sessione di elaborazione del testo.

La riga di stato di *emacs*

- Visualizza informazioni relative al testo corrente.
- Struttura:



- **St**: indica se il file è stato salvato dopo l'ultima modifica.
  "**" (non salvato), "--" (salvato), "%%" (file di sola lettura)
- **Nomebuffer**: indica il nome del buffer corrente
- **(major minor)**: modalità di editing del file.
  major fa riferimento a configurazioni di editing per linguaggi particolari (es. Lisp, C, testo semplice,etc.)
  minor fa riferimento a modalità di inserimento testo particolari
- **Linea**: numero di linea su cui è posizionato il cursore
- **Posizione**: posizione del cursore in relazione all'inizio del file

Comandi principali di *emacs*

- In Emacs i comandi vengono invocati attraverso la combinazione dei tasti CTRL o ALT con altri tasti.

  Ad esempio per uscire da Emacs si può usare la sequenza CTRL-x CTRL-c

Comandi di manipolazione dei file

`CTRL-x CTRL-f`	apre un file esistente
`CTRL-x CTRL-s`	salva il file corrente
`CTRL-x CTRL-w`	salva il file con nome

Comandi principali di *emacs* (2)

Comandi di manipolazione dei buffer

`CTRL-x b`	seleziona un buffer attivo o crea un buffer nuovo
`CTRL-x CTRL-b`	elenca i buffer attivi
`CTRL-x k`	elimina un buffer

Comandi di manipolazione delle finestre

`CTRL-x o`	seleziona un'altra finestra tra quelle attive
`CTRL-x 0`	chiudi la finestra corrente
`CTRL-x 1`	chiudi tutte le finestre eccetto quella corrente
`CTRL-x 2`	divide la finestra del buffer corrente in 2 (vert.)
`CTRL-x 3`	divide la finestra del buffer corrente in 2 (orizz.)
`CTRL-v`	scorrimento del testo in avanti
`ALT-v`	scorrimento del testo all'indietro

Comandi principali di *emacs* (3)

Comandi di spostamento del cursore

`CTRL-a`	sposta il cursore a inizio riga
`CTRL-e`	sposta il cursore a fine riga
`CTRL-b`	sposta il cursore a sinistra di 1 carattere
`CTRL-f`	sposta il cursore a destra di 1 carattere
`CTRL-n`	sposta il cursore alla riga sottostante
`ESC 6 CTRL-b`	cursore a sinistra di 6 caratteri
`ESC <`	sposta il cursore a inizio buffer
`ESC >`	sposta il cursore a fine buffer

Comandi principali di *emacs* (4)

Comandi di selezione di blocchi

`CTRL-barra spazio`	segna l'inizio del blocco
`ESC h`	definisce come blocco il paragrafo corrente
`CTRL-x CTRL-p`	definisce come blocco la pagina
`CTRL-w`	marca fine blocco e taglia
`ESC w`	copia un blocco in un buffer di memoria

Comandi principali di *emacs* (5)

Comandi di cancellazione

`CTRL-d`	cancella il carattere a destra del cursore
`BACKSPACE`	cancella il carattere a sinistra del cursore

Comandi di cancellazione con memorizzazione

`CTRL-k`	cancella la parte della riga a destra del cursore
`ESC d`	cancella parola dopo il cursore
`ESC BACKSPACE`	cancella parola prima del cursore
`CTRL-y`	inserisce dopo il cursore il testo cancellato
`CTRL-_`	annulla il comando precedente

Comandi di Undo

```
CTRL-x u
CTRL-_
```

Esempi

- Esempio: Sequenza che sposta la riga corrente in alto di tre righe.

```
CTRL-k
ESC 3 CTRL-p
CTRL y
```

- Esempio: Operazioni su paragrafi.

```
ESC h CTRL-w          cancella un paragrafo
ESC h ESC w           copia un paragrafo
```

- Esempio: Copia e incolla. Sequenza che copia la pagina corrente nel buffer di memoria e la incolla all'inizio del file.

```
CTRL-x CTRL-p ESC w
ESC <
CTRL-y
```

Comandi principali di *emacs* (6)

Comandi di ricerca di stringhe

`CTRL-s`	cerca un stringa in avanti
`ESC CTRL-s`	cerca un'espressione regolare in avanti
`ESC x replace-string`	esegue una sostituzione globale
`ESC x replace-regexp`	esegue una sostituzione con espressioni regolari
`ESC %`	esegue una sostituzione condizionale (query-replace)
`ALT-x occur`	cerca un'espressione regolare e salva il risultato

Esempio

```
ESC % stringa_1 [Invio] stringa_2 [Invio] opzione
```

- Questa sequenza di comandi permette di sostituire le occorrenze nel testo di **`stringa_1`** con **`stringa_2`**, con una procedura interattiva. Dopo il secondo comando di [Invio], all'utente viene chiesto di selezionare per l'occorrenza corrente un'opzione fra le seguenti:
  - Y o barra spazio

    Sostituisce e passa alla prossima occorrenza
  - N o Canc

    Non sostituisce e passa alla prossima
  - ^

    Salta all'occorrenza precedente
  - .

    Sostituisce l'occorrenza ed esce

Ambiente shell

Testo di riferimento:

V. Manca

“Metodi Informazionali”

Bollati Boringhieri

La shell

- Mezzo principale tramite il quale l'utente può interagire con il computer.
- Offre un insieme di funzionalità che costituiscono un ambiente operativo che permette all'utente di lavorare.
- Linux ha diversi tipi di shell
  - Shell di riferimento: **`bash`**

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
  - inizia con una parola chiave
  - può avere uno o più argomenti
  - viene chiusa da un ritorno a capo o da **;**

  (Eccezione: istruzione di assegnamento)

  Esempi:

```
$ echo esempio di echo
esempio di echo
$
```

```
$ echo esempio1;  echo esempio2
esempio1
esempio2
$
```

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
  - inizia con una parola chiave
  - può avere uno o più argomenti
  - viene chiusa da un ritorno a capo o da ;

(Eccezione: istruzione di assegnamento)

Esempi:

```
$ echo esempio di echo
esempio di echo
$
```

```
$ echo esempio1; echo esempio2
esempio1
esempio2
$
```

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
  - inizia con una parola chiave
  - può avere uno o più argomenti
  - viene chiusa da un ritorno a capo o da ;

  (Eccezione: istruzione di assegnamento)

  Esempi:

```
$ echo esempio di echo
esempio di echo
$
```

```
$ echo esempio1; echo esempio2
esempio1
esempio2
$
```

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
  - inizia con una parola chiave
  - può avere uno o più argomenti
  - viene chiusa da un ritorno a capo o da ;

  (Eccezione: istruzione di assegnamento)

  Esempi:

```
$ echo esempio di echo
esempio di echo
$
```

```
$ echo esempio1;  echo esempio2
esempio1
esempio2
$
```

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
  - inizia con una parola chiave
  - può avere uno o più argomenti
  - viene chiusa da un ritorno a capo o da ;

  (Eccezione: istruzione di assegnamento)

  Esempi:

```
$ echo esempio di echo
esempio di echo
$
```

```
$ echo esempio1 ; echo esempio2
esempio1
esempio2
$
```

Istruzioni e variabili della shell (2)

- Le istruzioni si appoggiano sulle variabili per poter fare le loro elaborazioni.
- Una **variabile** è un recipiente atto a contenere dei dati che possono essere input o output di una istruzione.
- Una variabile è identificata da un **nome**, il quale:
  - non può contenere caratteri speciali (?, *, ecc.).
  - è case-sensitive, cioè maiuscole e minuscole sono diverse.
  - deve iniziare con una lettera o con underscore (_)
- Solitamente il concetto di variabile va a pari passo con quello di **tipo di dato** che la variabile andrà a contenere.
- In ambiente bash, le variabili possono essere solo di tipo **stringa**, ovvero il loro contenuto è sempre una sequenza di caratteri.

Assegnamento di una variabile

- Per inserire un valore in una variabile si usa l'istruzione di **assegnamento**, che corrisponde al simbolo '='. Non inserire spazi tra il nome della variabile, l'uguale e il valore da inserire.

```
$ VARIABILE1=valore1
```

```
$ VARIABILE1 = valore1
```

Errore!

- Se il valore da dare contiene uno spazio è indispensabile racchiudere il valore tra doppi apici (es: `"valore con spazi"`)
- Una variabile viene creata al momento del suo primo assegnamento e rimane in memoria fino a che la shell rimane attiva.

Assegnamento di una variabile (2)

- Una variabile può assumere il valore speciale NULL, corrispondente a un valore indeterminato. Per assegnare tale valore si può scrivere

```
$ VARIABILE=
```

```
$ VARIABILE1=""
```

- Esempi:

```
$ VARIABILE1=valore1
$ VARIABILE2="valore 2"
```

Note sull'uso di una variabile

- Per **accedere al contenuto** di una variabile si utilizza il '**$**'. Questo permette di differenziare il semplice testo dal nome di variabili.
- Se si vuole accostare del testo a quello contenuto in una variabile è necessario **delimitare il nome della variabile** usando le graffe (es: **`${var}testo`**). L'uso delle { } ha come unico scopo quello di delimitare il nome della variabile.
- Per **vedere/stampare il contenuto** di una variabile si può usare il comando **`echo`**.

Esempio:

```
$ echo $VARIABILE2
valore 2
$ echo $VARIABILE1${VARIABILE2}000
valore1valore 2000
```

Variabili d'ambiente

- Le variabili normali sono visibili solo nella shell dove vengono dichiarate e il loro contenuto non è visibile dai processi lanciati dalla shell.
- Variabili d'ambiente
  - Possono essere associate ad un processo e sono visibili anche ai processi figli.
  - Possono essere usate per modificare il comportamento di certi comandi, senza dover impostare ripetutamente le stesse opzioni.
- Le variabili normali possono diventare variabili d'ambiente tramite l'istruzione **`export`**

  Esempio:

```
$ export VARIABILE1
$
```

Variabili d'ambiente (2)

- Quando ci si connette al sistema, alcune variabili d'ambiente vengono inizializzate con valori di default (modificabili solo dall'amministratore del sistema).
- Le principali variabili d'ambiente sono:
  - **`HOME`**: contiene il path assoluto della home dell'utente che ha fatto login.
  - **`MAIL`**: contiene il path assoluto di dove sono conenute le email dell'utente che sta usando la shell.
  - **`PATH`**: contiene la lista di directory, separate dai due punti, dove il sistema va a ricercare comandi e programmi.
  - **`MANPATH`**: lista di directory, separate dai due punti, per la ricerca delle pagine man da parte del comando man.
  - **`PS1`**: contiene la forma del prompt primario.
  - **`PS2`**: contiene la forma del prompt secondario.

Variabili d'ambiente (3)

- **`SHELL`**: contiene path assoluto e nome della shell in uso.
- **`TERM`**: contiene il nome che identifica il tipo di terminale in uso.
- **`LOGNAME`**: contiene il nome della login dell'utente che ha fatto login.
- **`PWD`**: contiene il path assoluto della directory corrente.

- L'utente può modificare a piacere il valore delle **proprie** variabili d'ambiente.

Variabili d'ambiente (4)

- Si può visualizzare la lista delle variabili d'ambiente con l'istruzione **`env`**

  Esempio:

```
$ env
HOME=/home/pippo
LOGNAME=pippo
MAIL=/var/spool/mail/pippo
...
$
```

Visualizzazione variabili

- Si può visualizzare la lista di tutte le variabili dichiarate nella shell con l'istruzione **`set`**

Esempio:

```
$ set
BASH=/bin/bash
BASH_VERSION=1.14.6(1)
...
HOME=/home/pippo
LOGNAME=pippo
MAIL=/var/spool/mail/pippo
...
SHELL=/bin/bash
TERM=linux
VARIABILE1=valore1
VARIABILE2=valore 2
$
```

Modalità di funzionamento shell

- La shell ha tre modalità di funzionamento:
  - **`Interattiva:`**
    La shell attende i comandi digitati dall'utente.
  - **`Di configurazione:`**
    La shell viene utilizzata per definire variabili e parametri d'utente e di sistema.
  - **`Di programmazione:`**
    La shell viene adoperata per ealizzare **procedure**, dette **script**, conententi costrutti di comandi/istruzioni di GNU/Linux.