# Modulo 2 - Appendice: L'editor *vi* e cenni su *sed*

**Laboratorio di Sistemi Operativi I**
**Anno Accademico 2006-2007**

Francesco Pedullà
(Tecnologie Informatiche)

Massimo Verola
(Informatica)

# Cos'è VI

- **Vi** è l'editor *storico* di UNIX ed è disponibile su tutte le piattaforme (AIX, HP/UX, Solaris) oltreché su Linux e su Windows
- E' piccolo: chiede poche risorse al sistema
- E' potente: ha un insieme di comandi completo e flessibile
- E' un editor di testo per cui produce solo file di testo puri a differenza, per esempio, di OpenOffice
- Esistono versioni *migliorate*:
  - Vim: aggiunge, tra l'altro, multi level undo, multi windows e buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection
  - Gvim: maggiore integrazione con l'interfaccia grafica (e.g., paste)
- **Emacs** è un altro editor tradizionale in ambiente UNIX/Linux, ancora più potente ma più complesso

# Cosa studiare

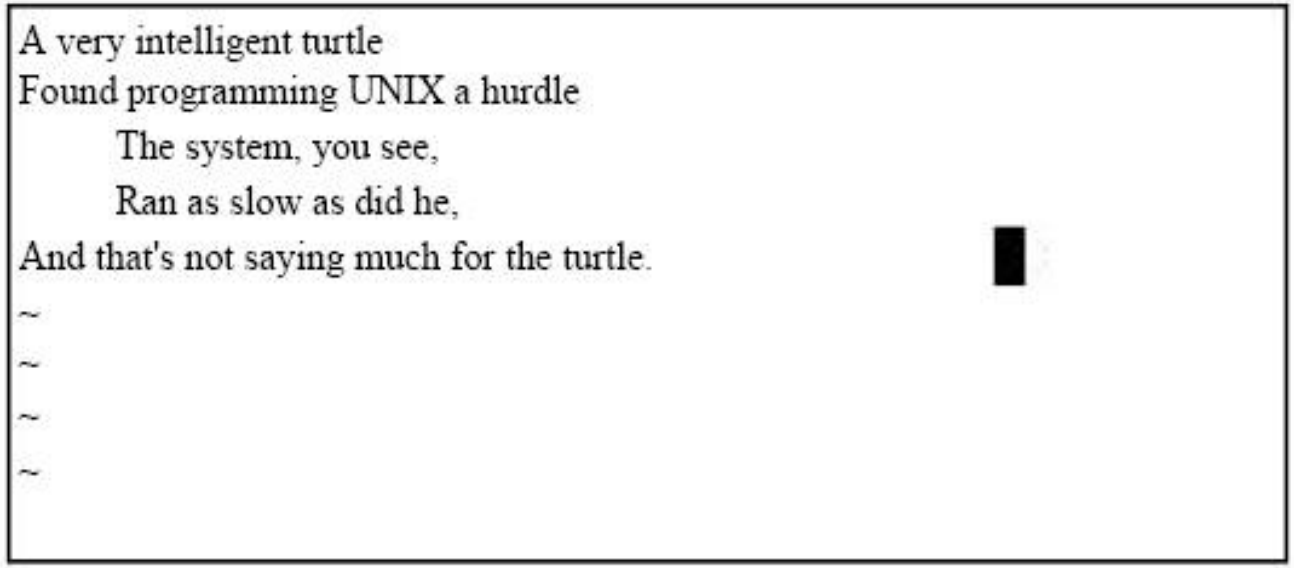
- Per approfondimenti potete usare:
  - Vimbook-OPL (reperibile all'URL http://truth.sk/vim/vimbook-OPL.pdf)
  - Vim-HOWTO (scaricabile da http://www.tldp.org)
  - Vimdoc (scaricabile da http://vimdoc.sf.net)
- Esistono molti altri tutorial disponibili su Internet che potete utilizzare
- Seguiremo soprattutto il primo testo (i primi tre capitoli)
- Il corso comprende solo questa lezione introduttiva che illustra i concetti fondamentali nell'uso di **`vim`**
- Una lezione non basta per usare bene **`vi`**: dovrete leggere uno dei documenti
- Ai fini dell'esame bisogna padroneggiare bene un editor di testo (vi o emacs)

# Aprire un file

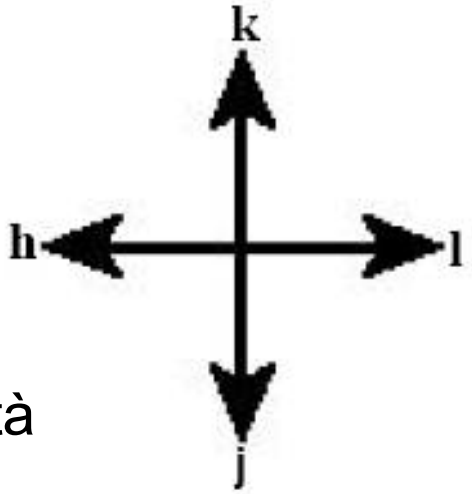
- Dalla linea comandi, nella directory in cui si vuole creare il file:
  - **`vim file.txt`**
- Il carattere ~ indica righe che non appartengono al file
- **`vim`** parte in modo *normale*
  - L'ultima riga mostra alcune informazioni utili
  - I caratteri digitati in modo normale non sono inseriti nel documento ma interpretati come comandi

# Inserire testo nel file

- Per passare in modo *inserimento* digitare **`i`**
- Scrivere il testo desiderato
- Si può cancellare il testo con il tasto **BACKSPACE**
- Per inserire una nuova linea digitare **RETURN**
- Inserito il testo, uscire dal modo *inserimento* premendo **ESC**



# Muoversi nel testo




- Per muovere il cursore abbiamo due possibilità:
  - Usare **`h,k,j,l`** per muoversi come in figura
  - Usare le frecce
- Per muovere il testo di una pagina abbiamo due possibilità
  - Usare **`Ctrl-u`** e **`Ctrl-d`** per salire o scendere
  - Usare i tasti **`PgUp`** e **`PgDn`**
- Per muovere il cursore lungo una linea di testo abbiamo due possibilità:
  - Usare **`0`** o **`$`** per andare all'inizio o alla fine della linea
  - Usare i tasti **`Home`** o **`End`**
- L'uso dei tasti speciali (frecce, etc.) è permesso sia in modo normale che in modo inserimento

# Modificare il testo

- Per cancellare un carattere abbiamo due possibilità:
  - Usare **`x`**
  - Usare **BACKSPACE**
- Per cancellare una linea (tutta o in parte):
  - Tutta la linea (ovunque sia il cursore): **`dd`**
  - Il resto della linea (a partire dal cursore): **`D`**
- Per inserire nuovo testo:
  - Nella posizione del cursore: **`i`**
  - Dalla riga successiva: **`o`**
  - Dalla riga precedente: **`O`**
  - Dal carattere successivo: **`a`**
  - Alla fine della linea: **`A`**
- Altri comandi utili
  - Undo: **`u`**
  - Redo: **`Ctrl-r`**

# Come terminare

- Bisogna andare in modo *comando* digitando il carattere **:**
- Per uscire dall'editor:
  - Se si vuole anche salvare il file, usare il comando **ZZ** (senza entrare in modo comando) oppure **:wq**
  - Se si vuole soltanto uscire, usare il comando :**q**
  - Se si vuole uscire senza salvare il file, usare il comando :**q!**
  - Se si vuole solo salvare il file, usare il comando :**w**
- Per uscire dal modo comando e tornare al modo normale usare **ESC**

# Riassunto dei modi e dei comandi principali

| Modo | Per entrare | A cosa serve |
|---|---|---|
| *Normale* | ESC | Muovere il cursore |
| *Inserimento* | i | Inserire il testo |
| *Comando* | : | Salvare/chiudere il file |

| Modo | Comando | Funzione |
|---|---|---|
| *Normale* | h,k,j,l | Spostamento del cursore (un carattere) |
| | Ctrl-u, Ctrl-d | Cambio pagina |
| | 0, $ | Spostamento del cursore (inizio o fine linea) |
| | X | Cancellazione di un carattere |
| | dd, D | Cancellazione della linea (intera o in parte) |
| | u, Ctrl-r | Undo, repeat |
| *Inserimento* | i,o,O,a,A | Inserimento nel testo |
| | BACKSPACE | Cancellazione di un carattere |
| | PgUp, PgDn | Cambio pagina |
| | Home, End | Spostamento del cursore (inizio o fine linea) |
| | Frecce | Spostamento del cursore (un carattere) |
| *Comando* | ZZ | Esci e salva |
| | q | Esci |
| | q! | Esci senza salvare |

# Esercizio

- Creare due file, **usage.sh** e **max.sh**, contenenti rispettivamente:

```
#!/bin/bash
usage () {
   if [ -n "$1" ] ; then
      echo “Usage: " \
       "$1 [options] args”
   fi
}

# Prints the usage
usage $(basename $0)
```

```
#!/bin/bash
max () {
   if [ $1 -gt $2 ] ; then
      return $1
   else
      return $2
   fi
}

max 12 14
echo $?
```

# Ripetizione di comandi e help

- E' possibile ripetere automaticamente il comando, premettendo allo specifico comando un numero:
  - **`10x`** cancella 10 caratteri
  - **`5dd`** cancella 5 linee
  - **`3$`** muove il cursore alla fine della terza linea successiva
- Per accedere all'help online, andare in modo comando e digitare **`help`** oppure premere **`F1`**
- E' disponibile anche un tutor con il comando (dalla shell Linux!) **`vimtutor`**

# Altri comandi di movimento

- Sono disponibili comandi di movimento più sofisticati:
  - Per avanzare (o indietreggiare) di una parola, usare il comando **`w`** (o **`b`**): **`5w`** sposta il cursore 5 parole più avanti
  - Per cercare un carattere sulla linea corrente, usare il comando **`f`** (o **`F`** verso sinistra): **`3fy`** ricerca la terza "y" nella linea del cursore
  - Per andare alla linea numero **`n`** (inizio prima parola) usare il comando n**`G`** (**`G`** senza argomento va alla fine del file)
- Per conoscere la posizione del cursore nel file: **`Ctrl-g`**
- Per vedere i numeri di linea del testo usare il comando **`:set number`** (per eliminarli **`:set nonumber`**)

# I comandi `d, c, y, p`

- Il comando **d** (delete) può essere seguito da un comando di movimento e cancella il testo dal cursore fino al testo trovato:
  - **dw** cancella fino alla fine della parola
  - **d3w** cancella le tre parole successive
  - **d$** cancella fino alla fine della linea (come i comandi **dd** e **D**)
- Anche il comando **c** (change) può essere seguito da un comando di movimento
  - **cw** cambia la parola successiva
  - **c$** cambia il testo fino alla fine della linea (come i comandi **cc** e **C**)
- Lo stesso vale per **y** (yank)
  - **yw** copia la parola seguente nel buffer
  - **yy** copia tutta la linea
- Il comando **p** (put) inserisce il contenuto del buffer a partire dalla posizione del cursore

# Altri comandi di modifica

- Il comando **`J`** (join) unisce la linea corrente e quella successiva:
  - **`3J`** unisce tre linee
- Il comando r (replace) sostituisce il carattere dove è posizionato il cursore
  - **`rx`** sostituisce il carattere con **`x`**
  - **`3rx`** sostituisce tre caratteri con **`x`**
- Il comando **`R`** sostituisce il testo finché non si esce dal modo di inserimento
- Il comando **`~`** scambia minuscole con maiuscole (e viceversa)
  - **`5~`** scambia maiuscole con minuscole nei 5 caratteri successivi
- Il comando **`.`** ripete l'ultimo comando di modifica

# Ricerca

- Per cercare una stringa usare il comando **/**
  - **/abc** ricerca abc a partire dal cursore
- Alcuni caratteri nella stringa di ricerca hanno un significato speciale:
  - **$** indica la fine della linea
  - **^** indice l'inizio della linea
  - **.** indica un qualunque carattere
  - Altri caratteri speciali: *** [ ] \ ? ~ / %**
  - Se la stringa cercata contiene un carattere speciale, questo va preceduto da \
- Il comando **?** esegue la ricerca all'indietro
- Il comando **n** ripete la ricerca

# Le macro - I

- Supponiamo di voler modificare un file come segue:

```
stdio.h
fcntl.h
unistd.h
stdlib.h
main()
{
...
```

```
#include "stdio.h"
#include "fcntl.h"
#include "unistd.h"
#include "stdlib.h"
main()
{
...
```

# Le macro - II

- Registriamo una macro che modifica come richiesto una singola riga, e poi la ripetiamo per ogni riga successiva
  - **qa** inizia la registrazione nel registro **a** (i registri vanno da **a** a **z**)
  - **^** va all'inizio della linea
  - **i#include"<ESC>** inserisce il testo prima del nome del file
  - **$** va alla fine della linea
  - **a"<ESC>** aggiunge il carattere " alla fine della linea
  - **j** va alla linea successiva
  - **q** finisce la registrazione
- La macro può essere invocata con **@a** (con **3@a** l'intera modifica è fatta!)

# Esercizi

- Aprire un file di testo con **`vi`** ed eseguire le seguenti operazioni:
  - inserire una nuova linea di testo in fondo al file;
  - copiare le ultime 4 linee del file all'inizio del file;
  - sostituire tutte le occorrenze della stringa *are* con il carattere - ;
  - salvare le modifiche.
- Aprire un file con **`vi`** e ripetere l'esempio della macro dopo avere inserito il testo su cui intervenire

# sed (I)

- Il nome del comando **`sed`** sta per **S**tream **ED**itor e permette di editare un testo, per esempio letto da stdin in una pipeline oppure da un file.
- La sua sintassi è la seguente: **`sed actions files`**
- **`actions`** è un'espressione composta dagli indirizzi di linea da un'azione, generalmente indicata da un carattere, da svolgere sugli indirizzi specificati (per la corretta sintassi, che infatti risulta molto simile a quella dei comandi accettati da **`vi`**, si veda **`info sed`**).
- Le **`actions`** possono manipolare il testo in vario modo:
  **`i`** = inserisci testo, **`s`** = trova e sostituisci, **`y`** = trasforma caratteri,
  **`d`** = cancella, **`p`** = stampa, **`q`** = esci da sed, ...
- Se **`actions`** è una stringa vuota (' '), **`sed`** stampa sullo standard output le linee in input, lasciandole inalterate.

# sed (II)

- Gli indirizzi di linea si specificano come numeri o espressioni regolari:
  '4' quarta linea
  '4,8' dalla quarta all'ottava linea (incluse)
  '4,$' dalla quarta all'ultima linea
  '1~2' a partire dalla prima linea, ogni 2 linee (cioe' le linee dispari)
  '/sh/' tutte le righe che contengono la stringa *'sh'*
  '4,/sh/' dalla quarta alla prima linea che contiene la stringa *'sh'* (inclusa)
- Se viene specificato un singolo indirizzo, l'azione viene svolta sulle linee che fanno il *match* con quella specifica di indirizzo; se viene specificato un intervallo, l'azione viene svolta su tutte le linee incluse nell'intervallo (estremi inclusi)
- Se non viene specificato un indirizzo o un intervallo di indirizzi di linea su cui eseguire l'azione, quest'ultima viene applicata a tutte le linee in input.
- Se vi è più di un'azione, esse possono essere specificate sulla riga di comando precedendo ognuna con l'opzione **-e**, oppure possono essere lette da un file esterno specificato sulla linea di comando con l'opzione **-f**.

# Esempi d'uso di sed

- **`> sed '4,$d' /etc/passwd`**
  stampa a video soltanto le prime 3 righe del file **`/etc/passwd`**:
  **d** è il comando di cancellazione che elimina dall'output tutte le righe a partire dalla quarta (si noti l'uso del quoting per impedire che la shell interpreti il metacarattere $).

- **`> sed 3q /etc/passwd`**
  stesso effetto del precedente comando: in questo caso **sed** esce dopo aver elaborato la terza riga (**3q**).

- **`> sed /sh/y/:0/_%/ /etc/passwd`**
  sostituisce in tutte le righe che contengono la stringa **sh** il carattere **:** con il carattere **_** ed il carattere **0** con il carattere **%**.

- **`> sed '/sh/!y/:0/_%/' /etc/passwd`**
  sostituisce in tutte le righe che non contengono la stringa **sh** il carattere **:** con il carattere **_** ed il carattere **0** con il carattere **%** (si noti l'uso del quoting per impedire che la shell interpreti il metacarattere **!**).

# Sostituzione del testo con sed

- Il formato dell'azione di sostituzione in **sed** è il seguente
  **s/expr/new/flags**
  dove:
  **expr** è l'espressione da cercare,
  **new** è la stringa da sostituire al posto di **expr**,
  **flags** può assumere una delle seguenti forme:
  - un numero da 0 a 9 che specifica quale occorrenza di expr deve essere sostituita (di default è la prima),
  - **g**: (global) ogni occorrenza di **expr** viene sostituita,
  - **p**: (print) la linea corrente viene stampata sullo standard output nel caso vi sia stata una sostituzione,
  - **w *file***: (write) la linea corrente viene accodata nel file nel caso vi sia stata una sostituzione.

# Esempi di sostituzioni con sed

- **`sed '/^root/,/^bin/s/:.............:/::/w disabled.txt' /etc/passwd`**
  Nelle righe del file **`/etc/passwd`** comprese fra quella che inizia con **`root`** e quella che inizia con **`bin,`** sostituisce la password criptata (lunga 13 caratteri) con la stringa vuota; tali righe sono poi accodate nel file *disabled.txt*.

- **`cat /etc/passwd | sed 's?/bin/.*sh$?/usr/local&?'`**
  Cerca tutte le righe in input in cui compare la stringa corrispondente all'espressione regolare **`/bin/.*sh$`** (ad esempio **`/bin/bash`**) e sostituisce quest'ultima con la stringa corrispondente a **`/usr/local/bin/.*sh$`** (ad esempio **`/usr/local/bin/bash`**).
  Si noti che, siccome il carattere separatore di sed compare nella stringa da cercare, si è usato il carattere **`?`** come separatore.
  Inoltre il carattere **`&`** viene rimpiazzato automaticamente da **`sed`** con la stringa cercata (corrispondente a **`/bin/.*sh$`**).

# Esercizi con sed e vi

- Ripetere gli esempi di sostituzione e ricerca con **`sed`** aprendo con **`vi`** il file di input e utilizzando i comandi di **`vi`**
- Utilizzando l'help di **`vi`**, verificare quali sono le opzioni dei comandi di sostituzione e ricerca
- Utilizzando **`sed`** sostituire nel file di input tutte le cifre da 0 a 9 con **`-`**