

Editor di testo

Ci serve un editor di testo ?

- Cambiare una semplice configurazione ?

```
$ echo "1" > /etc/sys/net/ipv4/ip_forward
```

- Aggiungere una linea alla fine di un file ?

```
$ echo "x11-vm/compiz ~x86" >> /etc/portage/package.keywords
```

- Configurare /etc/apache2/httpd.conf ?

Ci serve decisamente un editor di testo =)

Editor Wars : vi vs EMACS

VI

- Editor Modale
- Piccolo e leggero
(fa una cosa ma bene)
- Ambiente testuale

EMACS

- Uso estensivo delle combinazioni di tasti
- “Bells andr Whistle”
- GUI

Entrambi hanno una curva di apprendimento relativamente alta

vi appartiene allo standard POSIX (IEEE 1003 – ISO/IEC 9945)

Non mi è mai mancato nulla usando..



- gedit
- kate / kwrite
- nano
- joe
- oowriter
- ...
- notepad (>_>")

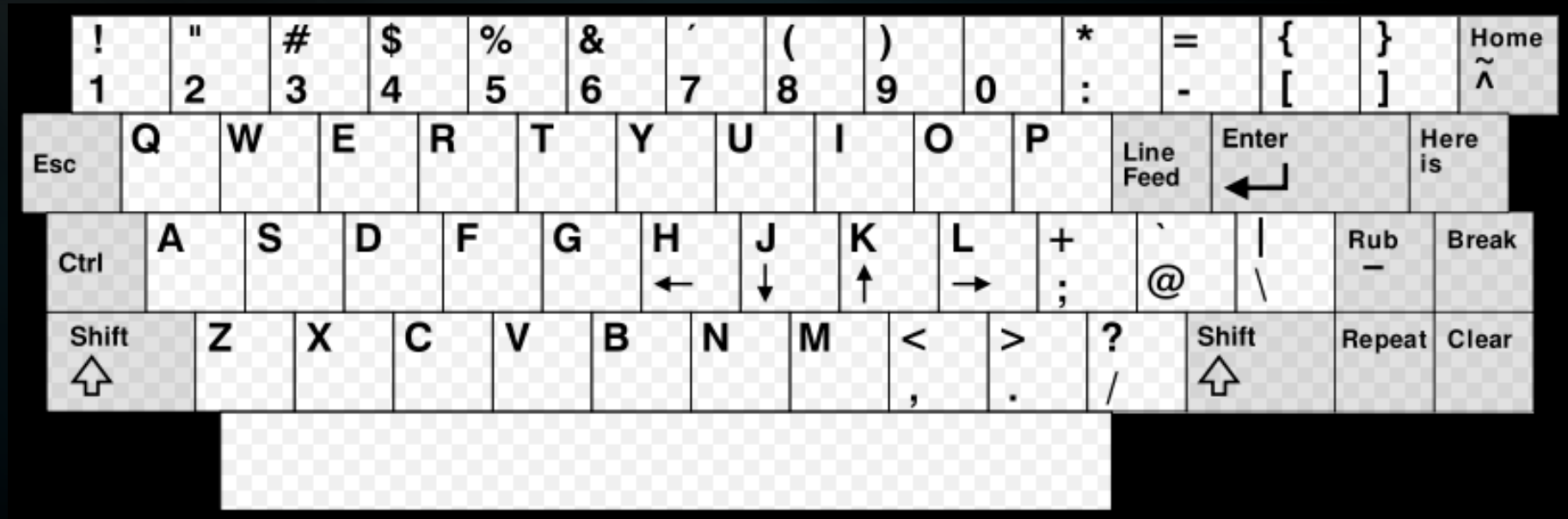
Perché imparare ad usare vi ?

- E' un editor prestante
 - Compiti complessi con pochi tasti
 - Efficace supporto per le regex nelle search and replace
- E' presente in ogni sistema certificato POSIX
 - come Mac Os X, Solaris, BSD/OS, HP-UX, AIX, ...
 - e Microsoft Windows including SFU (Services For Unix)
 - ovvero Win2003Server, Windows Vista Enterprise / Ultimate
- E' il miglior investimento dopo il touch-typing
- Aumenta la produttività
 - quando siete padroni dell'editor, l'attività di scrittura richiede meno tempo, lasciandovi più tempo per riflettere.

Vi : Jargon

- vi: /V-I/, not, /vi:/, never, /siks/, n.
[from 'Visual Interface'] A screen editor crufted together by Bill Joy for an early BSD release. [...]
- Tends to frustrate new users no end [...]

ADM3A Keyboard layout



<Esc> si trovava dove ora c'è <Tab>

<h>, <j>, <k>, <l> erano usati anche come cursori

Vi Clones

Ci sono diverse versioni (o Cloni) di vi, ognuno con funzionalità aggiuntive, ma che condividono tutti lo stesso sottoinsieme di comandi e meccanismo di funzionamento.

- Vim (aka "Vi IMproved") <http://www.vim.org>
- Elvis
- nvi
- Vile
- Gvim (with gui)
- Bvi (for binary files)
- Calvin (for DOS OS)
- Lemmy, Elwin , WinVi (for Windows OS)
- ...

“Send in the clones !”

Entering vi

Diversi modi di avviare vi :

```
you@yourhost$ vi
you@yourhost$ vi file           # Crea file se non esiste
you@yourhost$ vi file1 file2 ... # Visualizza il primo [...]
you@yourhost$ vi -r file       # Recupera file dopo un crash
you@yourhost$ vi +string file  # Esegue il comando string
    such as $ vi +23 file     # Apre il file alla linea 23
you@yourhost$ vi @rcfile      # Legge i comandi da @rcfile
```

Convenzione : “ <x> “ significa “ Press the 'x' key “

Provate :

```
<a> Hello folks! <Esc><:><w><q><Enter>
```

Poi :

```
you@yourhost$ cat file
```

Vi Modes

- **Command mode**

Il cursore è sul testo, potete digitare solo comandi, e nessuno dei caratteri che digitate viene visualizzato. Potete solo vedere gli effetti dei comandi che inserite.

- **Input mode**

Ci si aspetta che in questa modalità digitiate del testo, che viene visualizzato ed inserito. Usate questa modalità SOLO per inserire del nuovo testo.

- **Directive mode**

Vi si accede con il comando <:>.

Il cursore si sposta sull'ultima linea dello schermo preceduto da ':'
Vengono visualizzati i caratteri delle direttive inserite.

Switching modes

- **Command mode e insert mode**

Quando entrate in insert mode vedrete “ --INSERT-- “ in fondo a sinistra sullo schermo. Se non è presente è probabile che siate in command mode.
- **Potete accedere all'insert mode con <i> ...**

Ciò che digiterete verrà inserito prima della posizione del cursore.
- **... o con <a>**

'append' ha l'effetto di inserire il testo digitato dopo la posizione corrente del cursore.
- **<Esc> permette di tornare in modalità normale**

<:> apre *directive line* dove si possono inserire comandi per la gestione dei file come <w> per 'write' e <q> per 'quit'.
quindi, <a> [text] <Esc><:><w><q>
significa “ aggiungi *text* dal cursore, poi salva ed esci da vi”

Directive line : comandi base

: w	salva il file corrente
: w <i>file</i>	salva il file corrente con nome <i>file</i>
: w!	salva il file corrente sovrascrivendo
: q	esce dall'editor
: q!	esce dall'editor senza salvare cambiamenti
: wq	esce dall'editor salvando i cambiamenti
: x	esce dall'editor salvando i cambiamenti
: e <i>file</i>	'edit' apre <i>file</i>
: r <i>file</i>	'read' copia il contenuto di <i>file</i> sulla linea successiva
: n	'next' passa al prossimo file (se avete lanciato vi con \$ vi file1 file2 ...)
: rew	'rewind' file list, torna al primo file della lista
: rec <i>file</i>	'recover' recupera <i>file</i>
: split <i>file</i>	apre <i>file</i> in un altro buffer
: ab <i>tok exp</i>	crea una macro per <i>exp</i>
: unab <i>tok</i>	elimina una macro associata a <i>tok</i>
: n	va alla linea <i>n</i>

Spostare il cursore

<h> , <j> , <k> , <l> MOVE LEFT, MOVE DOWN, MOVE UP, MOVE RIGHT

G	Muove all'inizio del testo
H	Muove all'inizio dello schermo
M	Muove al centro dello schermo
L	Muove alla fine dello schermo
{ or }	Muove all'inizio o alla fine del paragrafo
(or)	Muove all'inizio o alla fine della frase
\$	Muove alla fine della linea
^	Muove al primo carattere non-blank della linea
n	Muove alla colonna <i>n</i> della linea corrente
w	Muove al primo carattere della prossima parola
e	Muove all'ultimo carattere della prossima parola
b	Muove al primo carattere della parola precedente
+	Muove al primo carattere non-blank della prossima linea
-	Muove al primo carattere non-blank della linea precedente
fx	Muove al primo carattere <i>x</i> trovato

*n*C esegue il comando *C* *n* volte

Entrare nella modalità inserimento

Ctrl+G	visualizza informazioni inerenti il file corrente sull'ultima linea nella forma <i>"filename" #lines %displayed</i>
i	entra in insert mode, prima della posizione corrente
I	entra in insert mode, all'inizio della linea
a	entra in insert mode, dopo la posizione corrente
A	entra in insert mode, alla fine della linea
o	insert mode in una nuova linea, sotto la linea corrente
O	insert mode in una nuova linea, sopra la linea corrente
J	unisce la linea corrente e la successiva
u	annulla l'ultimo cambiamento
U	annulla tutti i cambiamenti effettuati sulla linea
<<	sposta a sinistra la linea corrente
>>	sposta a destra la linea corrente
<L	sposta a sinistra tutte le linee dalla corrente all'ultima visualizzata
>L	sposta a destra tutte le linee dalla corrente all'ultima visualizzata

Basic commands: yank, delete, paste

y# 'yank' copia dal cursore alla fine del range specificato da #
yy copia la linea corrente
yw copia fino al primo carattere della prossima parola
y) copia fino alla fine della frase corrente
y} copia fino alla fine del paragrafo corrente
yfz copia fino al primo carattere 'z' trovato
y3j copia 3 linee a partire dalla corrente
y\$ copia fino alla fine della linea corrente

d# 'delete' dal cursore alla fine del range specificato da #

dd , dw , d) , d} , dfz , d3k , d\$... come sopra

c# 'change' taglia dal cursore alla fine del range #,
ed entra in insert mode nella posizione successiva.

p copia il contenuto del buffer nella posizione corrente
P copia il contenuto del buffer dopo la posizione corrente

... notate nulla ?

Combinare I comandi

- In vi si possono combinare comandi per svolgere compiti complessi, rapidamente mediante la pressione di pochi tasti.

Due comandi molto utili :

% identifica un blocco (può essere usato come parametro per I comandi precedenti) a partire dal cursore

. Il comando 'dot' semplicemente ripete l'ultimo comando eseguito.

Si noti che : *<i> [text typed] <Esc>* è anch'esso un comando.

Un esempio

Immaginate di aver dichiarato tre nuove funzioni nell'header, e di doverle implementare : il primo passo è la copia delle signatures :

```
bool check(int a, char c);  
int value(int strange);  
int failure(int my_process);
```

Per definire i blocchi con un comune editor dovrete :

- [*con il mouse o con le frecce*] Selezionare ed eliminare l ' ; ' (3 volte)
- [*tornare alla tastiera*] Digitare '{', 'Enter', '}'
- [*con il mouse o con le frecce*] Spostarvi alla prossima signature
- [*tornare alla tastiera*] ripetere il passo 2, poi il passo 3, poi il passo 2

@[-_ -]@

Con vi tutto questo si può fare con :

```
'A' <backspace> { <Enter> } <Esc> <j.j.>
```

Good way to Modal Editing

- Usare esclusivamente la modalità inserimento e tornare ai comandi solo se strettamente necessario è il modo sbagliato di usare un editor modale.

Bisognerebbe usare esclusivamente la modalità di comando, entrando in insert mode solo per i brevi periodi di tempo nei quali effettivamente si inserisce del testo.

Questo perché anche l'inserimento di una porzione di testo è un comando ed andrebbe impartito in modo 'transazionale'.

Per rileggere ciò che si è scritto è opportuno tornare in modalità comando, per navigare velocemente il testo, ed eventualmente sostituire una parola con `<cw>newword<Esc>` per poi salvare la modifica con `:w <Enter>` senza la necessità di smettere di leggere... l'operazione può eventualmente essere ripetuta, nel caso si incontrasse una parola da sostituire nello stesso modo, semplicemente impartendo il comando 'dot'.

Un altro esempio

Programmando capita di aver a che fare con espressioni complesse :

```
if (!my_st.enf && do_smtg( s.isopn(), s.isdmz) && ( a & b.c )){ ... }
```

Se si volesse estrarre la **function call** ed assegnarla ad una variabile locale, si può portare il cursore sul carattere 'd' ed usare `<c%>` 'change match' per tagliare l'intera chiamata a funzione ed entrare in insert mode, scrivere il nome della variabile, tornare in modalità comandi ed inserire una nuova riga sopra la corrente con `<Esc> <O>` , digitare `var = <Ctrl+R>` ; e tornare in modalità comandi con `<Esc>` .

```
<fdc%> var <Esc><O> var = <Ctrl+R> ; <Esc>
```

Esercizi

Che effetti producono I seguenti comandi ?

3J , 3dl , 3iHi<Esc>u , yf6 , dfyp , cfyp , llcfxy<Esc>hp

Che differenza c'è tra I comandi “4c(“ e “2c2(“ ?

Come trovare uno per uno I prossimi tre caratteri 'x' ?

Come sostituire alle prossime 2 frasi un “doh!” ?

Search

<i>fx</i>	<i>find</i> trova il prossimo carattere <i>x</i>
<i>tx</i>	<i>'till</i> scorre fino al prossimo <i>x</i> sulla linea
<i>Fx</i>	<i>find</i> trova il precedente carattere <i>x</i>
<i>Tx</i>	<i>'till</i> scorre fino al precedente <i>x</i> sulla linea
<i>/pattern [/]</i>	Ricerca la stringa <i>pattern</i>
<i>?pattern [?]</i>	Ricerca la stringa <i>pattern</i> a ritroso nel testo
<i>;</i>	Ripete l'ultima ricerca
<i>,</i>	Inverte l'ordine dell'ultima ricerca
<i>/ , ?</i>	Ricerca usando l'ultimo pattern

Espressioni regolari

Un'espressione regolare (*regex*) è una stringa, solitamente racchiusa tra delimitatori, che include wildcards, moltiplicatori e sequenze di escape, al fine di poter riconoscere un'intera classe di stringhe.

Sono molto comuni nei linguaggi di programmazione e scripting ...

Diverse sintassi nelle varie implementazioni :

GNU grep, perl, php, python, tcl, etc ...

<code>^</code>	Inizio della linea	<code>.</code>	ogni carattere
<code>\$</code>	Fine della linea	<code>*</code>	occorrenza 0- <i>n</i>
<code>\w</code>	Alpha-numeric	<code>\<</code>	inizio parola
<code>\W</code>	Non Alpha-numeric	<code>\></code>	fine parola
<code>re\?</code>	<i>re</i> , occorrenza 0-1	<code>....</code>	
<code>re\+</code>	<i>re</i> , occorrenza 1- <i>n</i>		
<code>re er</code>	<i>re</i> oppure <i>er</i>		

Search and replace

	... with regex, in directive line
<code>:range</code>	definisce un range (<i>n</i> , . , \$, %)
<code>:r s / exp / text /</code>	sostituisce il primo match di <i>exp</i> con <i>text</i>
<code>:r s / exp / text / c</code>	chiede conferma prima di sostituire
<code>:r s / exp / text / g</code>	sostituisce ogni match di <i>exp</i> con <i>text</i>
<code>:r s / exp / text / n</code>	sostituisce I primi <i>n</i> match di <i>exp</i> con <i>text</i>

Search & exec command

<code>: n , m g / exp / cmd</code>	ricerca tra le linee tra <i>n</i> ed <i>m</i> il pattern <i>exp</i> , ed esegue il comando <i>cmd</i> .
eg.	
<code>: 4,16g/exp/s//text/</code>	ricerca <i>exp</i> tra le linee 4 e 16, e sostituisce ogni occorrenza trovata con <i>text</i>
<code>: 5,20g/exp/d</code>	elimina ogni istanza di <i>exp</i> nelle linee dalla 5 alla 20

Opzioni di ricerca

- `:set ic` ricerca 'ignore case'
- `:set noic` disabilita 'ignore case'
si usa `/pattern\c` per una singola ricerca ic
- `:set hlsearch` 'highlight' dei risultati
- `:set nohlsearch` disabilita hlsearch
- `:set incsearch` mostra match parziali
- ...
- `:set [no]option` disabilita l'opzione

Esercizi

Che effetti hanno I seguenti comandi ?

`:% s / mr / sig / c`

`:$ s / \<\W.*\W \> / uhuh / g`

`:6, 18 g / ^.*x\+$ / dd`

Come effettuare le seguenti operazioni ?

Eliminare tutti I verbi in forma infinita

Cercare e sostituire previa conferma con 'ch' tutte le occorrenze di 'k'

More Directive line commands

- : map *key cmd* effettua il binding del comando *cmd* su *key*
- : unmap *key* rimuove l'assegnazione a *key*
- : [*n,m*]w [*filename*] scrive dalla linea *n* alla *m* [sul file *filename*]
- : !*cmd* esegue un comando UNIX
- : !! ripete l'ultimo comando UNIX eseguito
- : w !*cmd* salva il file corrente e ne passa il contenuto ad un comando UNIX *cmd* attraverso una pipe.
- : r !*cmd* legge l'output di un comando UNIX *cmd*
- : sh avvia una shell
- : args visualizza la lista dei file bufferizzati, racchiudendo il file corrente tra []
eg. “ file1 [file2] file3 file4 “

Configuring clones

Vim

```
:e ~/.vimrc           for Unix  
:e $VIM/_vimrc       for MS-Windows  
  
...  
  
:r $VIMRUNTIME/vimrc_example.vim
```

Considerazioni

- Non si impara studiando, si impara con la pratica
parliamo sempre e (quasi) solo di vi
- Vimtutor
- Una buona padronanza dell'editor consiste nella capacità di concepire comandi complessi in tempi brevi che realizzino il risultato desiderato.

Questions ?
/(and Answers)?/

How the vi editor would seem if it has been made by Microsoft

```
!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1
/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="eng">
<head>
  <meta http-equiv="Content-Language" content="en-us" />
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link rel="shortcut icon" href="/images/icons/hmfy.ico" />
  <title>HMFY - Geekdom, Projects, Anti-Boredom Freedom Fighters</title>
  <style type="text/css" media="all"><!--@import url( /css/main_v2.css );
  @import url("/css/frontpage.css");
--></style>
  <script type="text/javascript" src="/js/main.js"></script>

  <script type="text/javascript">
    <!--
      if (top.frames.length!=0)
        top.location=self.document.location;
      window.defaultStatus='HMFY: Geekdom, Projects, Anti-Boredom Freedom Fighters'

      //-->
    </script>
</head>
<body>

<table id="main" cellpadding="0" cellspacing="0" border="0" width="100%">
<tr>
  <td valign="top" width="140" class="sidebar">
<!-- Sidebar Begin -->
  <div class="sidehead">Projects</div>
  <div class="sidebar">
```

1,1

Top

Online References

Vi – Wikipedia

<http://en.wikipedia.org/wiki/Vi>

The Vi Lovers homepage

<http://thomer.com/vi/vi.html>

Why do those #?@! nutheads use vi ?

<http://www.viemu.com/a-why-vi-vim.html>

Vi Clones and HomePages

<http://www.saki.com.au/mirror/vi/clones.php3>

Vi Reference Manual

<http://drumlin.thehutt.org/vi/>

Regexp Syntax Summary

<http://www.greenend.org.uk/rjk/2002/06/regexp.html>

Books

O'REILLY - Learning the vi Editor, Sixth Edition

<http://www.oreilly.com/catalog/vi6/>